ENCICLOPEDIA PRACTICA DE LA

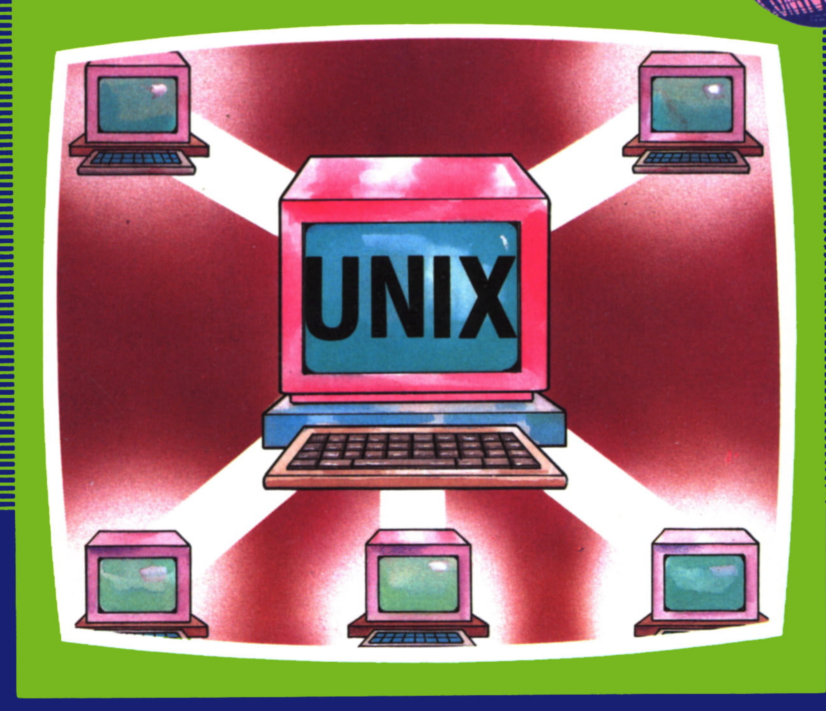
INFORMATICA APLICADA

18

UNIX, el estándar de los sistemas operativos multiusuario

Aula de Informática

SINCLAIR AMSTRAD COMMODORE IBM



EDICIONES SIGLO CULTURAL

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

18

UNIX, el estándar de los sistemas operativos multiusuario

EDICIONES SIGLO CULTURAL

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática
JOSE ARTECHE, Ingeniero de Telecomunicación

Diseño y maquetación:
BRAVO-LOFISH.

Dibujos:
JOSE OCHOA Y ANTONIO PERERA.

Tomo XVIII. **UNIX, el estándar de los sistemas operativos multiusuario**

AULA DE INFORMATICA APLICADA (AIA)
DOMINGO VILLASEÑOR CALVO, Diplomado en Informática
JESUS RONCERO GONZALEZ, Diplomado en Informática

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Sor Angela de la Cruz, 24-7.º G. Teléf. 279 40 36. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América. 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-061-8.
ISBN de la obra: 84-7688-018-9.

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M. 2.542-1987
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Sor Angela de la Cruz, 24-7.° G. Teléf. 279 40 36. 28020 Madrid

Marzo, 1987
P.V.P. Canarias: 365,-

INDICE

Los ejemplos de este libro han sido realizados en un miniordenador Factor (de Victory Computer Systems) propiedad de la Escuela Universitaria de Informática de la Universidad Politécnica de Madrid.

INTRODUCCION 1

¿QUE ES UN SISTEMA OPERATIVO?





N sistema operativo es un puente o intermediario entre la máquina o parte física (hardware), y el usuario (sus mandatos y peticiones). Cuando la instalación del ordenador es grande, lógicamente, y por necesidades económicas, los recursos del ordenador (tiempo de procesador, memoria, dispositivos periféricos, ...) se comparten entre varios usuarios (instalaciones multiusuario). El sistema operativo es el programa (o conjunto de programas) que permite a ese usuario o grupo de usuarios compartir estos recursos eficazmente. En definitiva, gestionar y optimizar el rendimiento de la máquina.

Podemos distinguir varios tipos de sistemas operativos:

— *Sistemas de proceso batch, o secuencial por lotes:* Son aquéllos que permiten ejecutar los trabajos uno a uno, de manera estrictamente secuencial. Su respuesta no es inmediata (horas, días...). Fueron el primer tipo de sistemas operativos.

— *Sistemas de multiprogramación:* Permiten que varios trabajos se puedan ejecutar simultáneamente.

— *Sistemas en tiempo real:* Deben permitir que el ordenador pueda recibir datos tan rápidamente como lleguen; en caso contrario, se perderán. La respuesta debe darse en un tiempo preestablecido.

— *Sistemas de tiempo compartido o «time sharing»:* Permiten a muchos usuarios utilizar el mismo ordenador, dando la apariencia de que sólo está dedicado a cada uno de ellos. Esto se consigue haciendo que el procesador central de la máquina atienda a cada usuario cíclicamente durante un determinado intervalo de tiempo.

UNIX pertenece a este grupo de sistemas operativos.



HISTORIA

UNIX es un producto que salió de los laboratorios Bell, filial de Western Electric Company, a su vez filial de AT&T. Todo empezó alrededor de 1968, cuando los laboratorios Bell abandonaron un proyecto que estaban desarrollando junto con otras entidades. Ese proyecto era un sistema operativo que no llegó nunca a realizarse: MULTICS.

Parte importante en el mismo era Ken Thompson, un programador de los laboratorios Bell, el cual tomó algunas de las ideas de MULTICS para concebir un sistema operativo de carácter interactivo como herramienta para la programación.

Entre 1968 y 1970 Ken Thompson escribió la primera versión de UNIX, que implementó sobre un miniordenador PDP-7 de Digital Equipment Corporation. Al poco tiempo se le unió Dennis Richie, junto con el cual escribía una nueva versión de UNIX para miniordenadores PDP-11, que por aquellos tiempos eran bastante populares.

En 1971, paralelamente con el desarrollo de los trabajos de UNIX, Dennis Richie trabajaba en la definición y en el compilador de un nuevo lenguaje: el «C». Poco después, en 1973, reescribían UNIX en lenguaje C (ya que antes estaba escrito en lenguaje ensamblador) para el ordenador PDP-11.

En 1974 organizaciones externas a los laboratorios Bell como las Universidades de Berkeley, en California, y de Columbia, obtuvieron, sin afán de lucro, licencias sobre UNIX, con lo cual por primera vez el sistema salía al exterior, y es en 1975 cuando UNIX se comercializa de manera definitiva.

En 1977 UNIX es transportado por primera vez a máquinas que no son PDP. Se producen modificaciones en el sistema y en el lenguaje C de cara a la portabilidad (facilidad de llevar programas de una máquina a otra diferente, sin efectuar cambios en los mismos). Estas modificaciones acaban en 1978, y UNIX se comercializa con el nombre de UNIX versión 7 (UNIX/V7).

Con el paso del tiempo el sistema va mejorando e incorporando nuevas utilidades. Así, en 1982 surge la versión UNIX System III, y en 1983 UNIX System V.

¿Cuál es la situación actual? Podemos considerar que la versión 7 y el Sistem III son las versiones más difundidas de UNIX, aunque AT&T ha dejado bien clara su intención de hacer del UNIX System V un estándar del mercado.



VARIANTES DE UNIX

A partir de que UNIX saliera de los laboratorios Bell y coincidiendo con la creciente popularidad y potencia de los microprocesadores, otras compañías y universidades han adaptado UNIX a otras máquinas, apareciendo diversas variantes del sistema básico. Estas variantes las podemos dividir en dos grupos:

— Por un lado, aquellos sistemas que corresponden a un transporte de UNIX sobre otra máquina, razón por la cual los laboratorios Bell perciben el pago de una licencia y de un canon por la difusión de los mismos. Entre éstos podemos citar el IS/1 (de Interactive Systems), el XENIX (de MicroSoft), Zeus (de Zilog) y UNIPLUS+ (de Unisoft Systems).

— Por otro lado, aquellos sistemas que «son como UNIX», pero escritos de nuevo o con tantas modificaciones que se les puede considerar distintos del producto de los laboratorios Bell y, por tanto, no necesitan pagar a éstos ningún tipo de licencia. Estos sistemas se podría decir que son «compatibles» con UNIX, y entre ellos cabe destacar el sistema SOL, desarrollado en Francia, y el IDRIS, desarrollado por Whitesmiths Ltd.



CARACTERISTICAS GENERALES

En un principio UNIX se pensó como un sistema operativo para miniordenadores y grandes equipos, pero el rápido avance habido en el campo de la tecnología electrónica y la aparición de microprocesadores cada vez más potentes, ha hecho que también penetre en el mundo de la microinformática, con lo cual hoy en día UNIX lo podemos encontrar en micros, minis y grandes ordenadores.

El sistema se divide en dos partes. La primera parte consiste en programas y servicios que han hecho de UNIX un entorno de programación tan popular. Es la parte realmente visible a los usuarios, incluyendo entre sus programas el intérprete de comandos (la shell), procesadores de texto (ntroff), editores (ed o vi), compilador de lenguaje C...

La segunda parte provee el soporte a estos programas y servicios en la máquina en concreto en que trabajemos.

Algunas de las características que hacen de UNIX uno de los sistemas operativos más potentes del mercado son las siguientes:

1. A diferencia de otros sistemas operativos que están escritos en lenguaje ensamblador de la máquina a que están destinados, *UNIX está escrito en un lenguaje de alto nivel, como es el C*, de ahí su alto grado de transportabilidad sobre distintas máquinas.

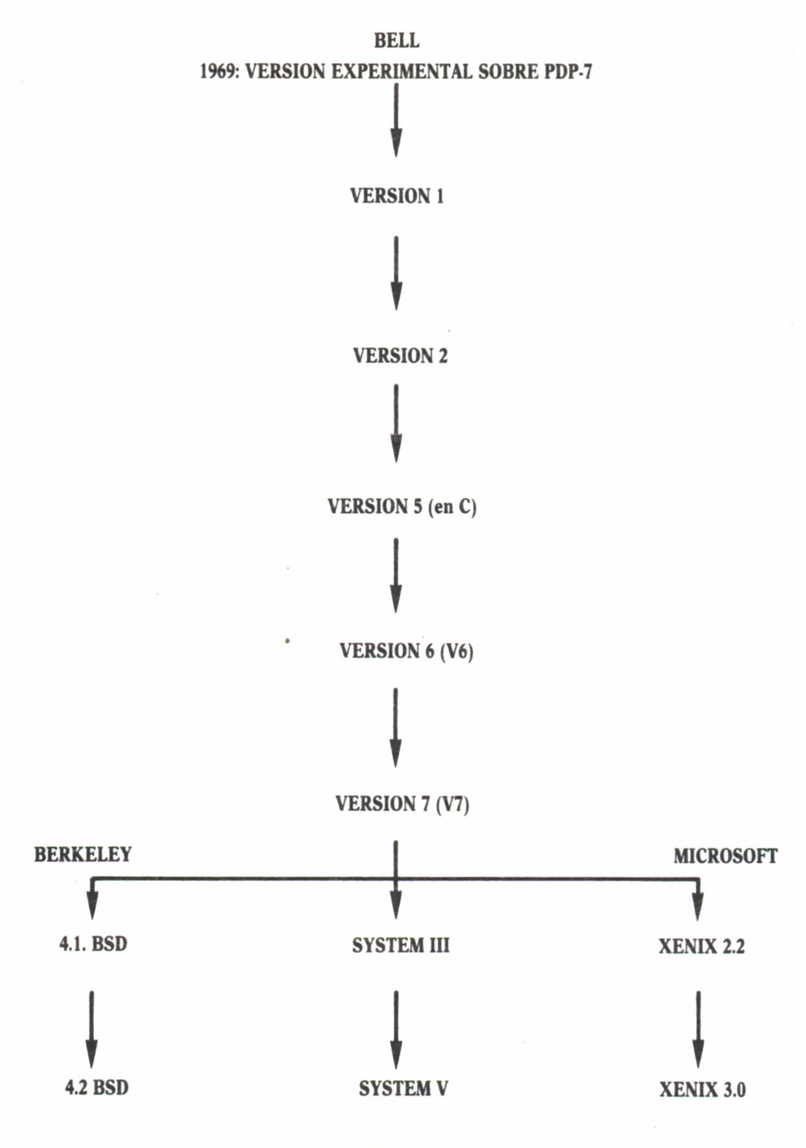


Fig. 1.1. Evolución de distintas versiones de UNIX.

2. *Tiene una estructura de ficheros jerarquizada,* es decir, en forma de árbol invertido. Este árbol nace en un directorio raíz que contiene otros subdirectorios y ficheros. Estos subdirectorios, a su vez, contienen otros subdirectorios y ficheros, y así sucesivamente.

3. *Redireccionamiento de la entrada/salida.* Esto quiere decir que la salida (el resultado) de un comando lo podemos llevar al sitio que queramos, bien la pantalla del terminal o bien a cualquier fichero en disco. Lo mismo sucede con la entrada de un comando, que puede tomar datos o bien del teclado o de cualquier fichero.

4. *La interface con el usuario* (la shell), que es lo que constituye el intérprete de comandos, *es fácilmente modificable,* con lo que se puede adaptar de manera que forme un entorno «amistoso» con el usuario, y haga transparente la complejidad de la sintaxis de UNIX. Esto quiere decir que se podría hacer un sistema de menús que fuera guiando al usuario para realizar las tareas necesarias sin que éste tenga que aprenderse ningún comando del sistema operativo.

5. *Es un sistema operativo multitarea,* es decir, que permite al ordenador ejecutar varias tareas o programas con aparente simultaneidad. Decimos aparente, porque al estar concebido UNIX sobre ordenadores de un solo procesador, es evidente que en un instante cualquiera de tiempo el procesador sólo puede estar atendiendo a un trabajo, pero esto es algo «transparente» al usuario, que cree estar ejecutando varios programas de manera simultánea.

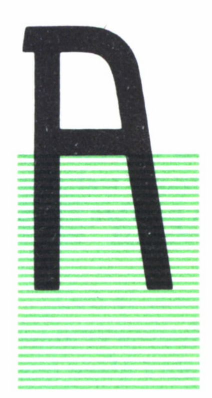
6. *Permite trabajar en modo multiusuario,* es decir, tiene capacidad para permitir el trabajo simultáneo de varios usuarios. Es misión del sistema operativo compartir los recursos del ordenador (CPU, memoria, dispositivos periféricos...) entre todos los puestos de trabajo conectados al sistema.

7. *Permite la comunicación entre procesos distintos* mediante pipe-lines («tuberías»). Esto significa que la salida de un proceso puede ser la entrada de otro, es decir, que podemos «enganchar» comandos para que la salida de uno sea la entrada del siguiente.

Todo lo visto anteriormente parecen ser alabanzas al sistema operativo UNIX. Pero también tiene algunos inconvenientes que no debemos olvidar, como son una mayor dificultad en su aprendizaje (muy superior a la de otros sistemas operativos como MS-DOS o CP/M), debido en gran parte al elevado número de comandos y a la engorrosa y a veces críptica sintaxis de los mismos, lo que hace de UNIX un sistema operativo casi inaccesible para el profano, aunque, como hemos dicho antes, este problema se puede solucionar modificando la «shell».

Otro inconveniente es el elevado número de versiones de UNIX que existen en el mercado, todas ellas con mayor o menor grado de incompatibilidad entre sí, lo cual hace difícil su estandarización. También, para solucionar esto, la compañía creadora de UNIX, AT&T, sacó al mercado la versión UNIX System V, que pretende ser el estándar dentro de la actual dispersión de versiones.

UNA PRIMERA TOMA DE CONTACTO 2



CONEXION AL SISTEMA: LOGIN



NTES de poder acceder al sistema UNIX debe verificarse que el terminal o puesto de trabajo desde el cual se va a operar está conectado. En caso de que esté usando un sistema independiente, consulte con el Manual del equipo.

Normalmente, una vez conectado el terminal ya dispone de la atención del ordenador, aunque en algunos equipos deberá pulsar antes la tecla RETURN o cualquier tecla similar.

Una vez dados estos pasos verá que en la pantalla aparece la palabra **«login:»** o también **«:login».** Esta palabra indica qúe se presente ante el sistema UNIX. Para ello debe disponer de un identificador de entrada o nombre de conexión, que permita al sistema comprobar si usted posee los derechos de acceso a los recursos y servicios prestados por la instalación a los usuarios. Normalmente este identificador lo proporciona el administrador del sistema.



CORRECCION DE ERRORES

Todo lo que se teclea se guarda en una zona de memoria (buffer de teclado) hasta que se pulsa la tecla RETURN. Si ha cometido algún error, puede corregirse mediante los siguientes convenios:

— Para borrar el último carácter tecleado se pone el carácter especial «#». Por ejemplo, supongamos que su identificador de entrada es «domingo». Un error se podría corregir así:

login : domib#ngo

Lo que se ha hecho ha sido borrar la «b» con el carácter «#». Aunque la «b» sigue en la pantalla, ya no es un carácter válido.

— Para borrar todo lo que se ha tecleado (se supone que todavía no se ha pulsado la tecla RETURN) se utiliza el carácter especial «@». Por ejemplo:

login : pedro @
domingo

Lo que se ha hecho ha sido borrar el nombre de conexión «pedro» entero y sustituirlo por «domingo», ya que antes se había supuesto que este último era el correcto.



USO DE LA PALABRA CLAVE

Una vez que el nombre de conexión es correcto se pulsa la tecla RETURN, con lo que se entra en el sistema y se podrá empezar a trabajar. Pero lo más usual es que antes el ordenador le pregunte la palabra clave mediante el mensaje:

Password :

Normalmente, una vez obtenido del administrador del sistema el nombre de conexión, se puede crear una palabra de paso o palabra clave que permitirá proteger sus ficheros, de manera que sólo usted tendrá acceso a ellos. Esta palabra clave debe ser secreta, y no la conoce ni tan siquiera el administrador del sistema, a menos que se divulgue. Es importante por este motivo que recuerde correctamente la palabra clave, ya que de lo contrario ni incluso usted podrá acceder a sus datos. De todas maneras, si se le ha olvidado el administrador puede anularla y se podrá crear otra nueva.

Una vez introducida la palabra clave (la cual no aparecerá en el terminal por motivos de seguridad), y si es correcta, nos ponemos bajo el control del shell (o la shell), que es el intérprete de mandatos de UNIX. El shell sacará por pantalla el carácter especial «$», el «%» u otro parecido. Ese carácter es el **prompt** (indicador) del sistema, y le indica que está preparado para trabajar y que espera alguna orden suya.

Un comando u orden que le puede ser útil es passwd, que sirve para crear o modificar su palabra clave:

```
%passwd
changing password for domingo
Old password:
New password:
Re-enter new password:
```

La palabra clave elegida no aparece en el terminal cuando la tecleamos. Con esta medida de seguridad se garantiza que sólo usted la conocerá, a menos que la divulgue. Para evitar posibles errores de escritura el sistema le pide por dos veces que introduzca la palabra clave. En caso de que no coincidan la primera y la segunda, no se dará por válida ninguna de las dos y el proceso empezará de nuevo.

```
:login: domingo
Password:
Login incorrect
login:
```



IDENTIFICACION DE CADA USUARIO

Con el comando **who** podemos ver quién está conectado al sistema en un momento determinado.

```
%who
f1457     tty02    Dec 23 16:56
e0254     tty03    Dec 23 17:19
%
```

La columna de la izquierda indica los nombres de conexión o identificadores de entrada, es decir, lo que se teclea cuando aparece **login:**. La siguiente columna indica los terminales o puestos de trabajo en que está cada usuario, y la columna de la derecha corresponde a la fecha y hora de conexión de cada uno.



CRITERIOS BASICOS PARA LA UTILIZACION DEL TERMINAL

Una vez dentro del sistema UNIX nos pueden ser útiles algunos criterios que nos facilitarán la utilización del terminal.

— *Detención de la ejecución de un programa.* Puede interrumpirse la ejecución de cualquier programa pulsando la tecla «DEL», «RUB» o «ctrl

+c». Cuando el sistema lo detecte visualizará de nuevo el prompt (normalmente el carácter «$» o el «%») y quedará a la espera de un nuevo mandato.

— *Detención del desplazamiento de texto en el terminal.* Hay veces en que al editar un texto o al ejecutar un comando, la salida que se obtiene excede de la capacidad visual de la pantalla (normalmente 25 filas), con lo que los caracteres van desapareciendo por la parte superior de la misma. Para poder evitar ese desplazamiento y poder examinar todo detenidamente se puede pulsar **«ctrl+s»** (teclas **ctrl** y tecla **s** simultáneamente) y **«ctrl+q»** para reanudar el desplazamiento.

— *Buffer de líneas de teclado.* El sistema puede leer la orden siguiente mientras está ejecutando la anterior, ya que los caracteres tecleados se almacenan en una zona de memoria especial (buffer de teclado). Si se teclea una orden que requiera algún tiempo en ejecutarse, antes de que aparezca el prompt del sistema indicando que ya está preparado para la siguiente orden puede teclearse otro comando, que aunque no aparezca en pantalla estará almacenado en el buffer de teclado. En cuanto el sistema acabe con lo que estaba haciendo aparecerá por pantalla lo que se tecleó anteriormente. Si era un comando y no llegó a pulsar RETURN, no se ejecutará nada.

— *Desconexión del sistema:* **logout.**

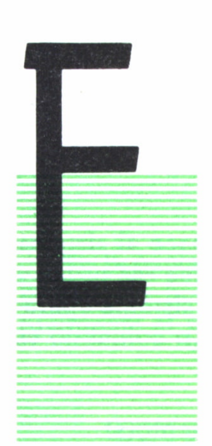
Cuando acabe una sesión de trabajo debe desconectarse del sistema UNIX. Esto se realiza pulsando **«ctrl+d».** Al efectuar esta operación no aparece ningún carácter en la pantalla. Sabrá que ha salido satisfactoriamente del sistema cuando en la pantalla aparezca de nuevo un mensaje de conexión (login:), que permitirá entrar a otros usuarios.

Recuerde, para salir del sistema *NO DESCONECTE EL TERMINAL.* La manera correcta de salir es pulsando «ctrl+d».

3 UNA INTERFACE ENTRE EL USUARIO Y EL SISTEMA OPERATIVO: EL SHELL



¿QUE ES EL SHELL?



EL shell es un programa que ayuda a la comunicación entre el usuario y el sistema operativo UNIX. Es un intérprete de órdenes. Traduce cada orden en una secuencia de acciones cuya finalidad es satisfacer las peticiones del usuario.

Inmediatamente después de la apertura de una sesión, el usuario se encuentra bajo el control del shell. El carácter especial (el *prompt*) que aparece en la pantalla al comienzo de la línea indica que está dispuesto a aceptar órdenes y, si son correctas, a ejecutarlas.

En el capítulo anterior dijimos que el prompt podía ser o el carácter especial «$» o el carácter «%». Ello es debido a que no hay un solo shell. El «$» es característico del Bourne Shell, que es el intérprete de órdenes que se suministra con la versión 7 del UNIX de los laboratorios Bell. El «%» es el prompt del «C» shell, escrito en la Universidad de Berkeley, en California.



CONCEPTO DE FICHERO

Un fichero es un conjunto de información al que se le asigna un nombre, y que está almacenado en un dispositivo de almacenamiento secundario como puede ser un disco o una cinta magnética. Esta información pueden ser programas, datos o textos.

Normalmente un fichero se crea con un editor de texto, como puede ser el «ed» o el «vi» (este último se verá más adelante). Una vez creado, permanece en el sistema hasta que se ordene al shell que lo elimine.

Para el sistema UNIX, un fichero no tiene ningún tipo de estructura interna; es simplemente una secuencia de caracteres que puede oscilar entre 0 y, aproximadamente, 1 billón, que es el máximo tamaño que puede

alcanzar un fichero en UNIX. El tamaño de los ficheros aumenta automáticamente al almacenar más información.

Un fichero se referencia por su nombre, que es una cadena de hasta catorce caracteres. Normalmente, en el nombre de un fichero se incluye tambien la extensión (aunque esto es opcional). La extensión de un fichero es una cadena de hasta tres caracteres precedida por un punto («.»). Sirve para indicar el tipo de información que tiene el fichero. Por ejemplo, un programa en lenguaje BASIC podría llamarse «práctica.bas», donde «práctica» sería el nombre propiamente dicho, y «.bas» sería la extensión. Así, los programas en COBOL podrían llevar de extensión «.cob», los programas en lenguaje Pascal «.pas», etc.



CONCEPTO DE COMANDO

Un comando es una orden o mandato que el usuario hace al sistema operativo. El intérprete de mandatos o shell es el que traduce esa orden y la desglosa en acciones más sencillas, a la vez que dirige al sistema operativo para que satisfaga la petición. La sintaxis general de un comando es la siguiente:

NombreDeComando   opción(es)   argumento(s)

— El nombre del mandato está representado por algunos caracteres. Debe explicar por sí mismo el tratamiento realizado (por ejemplo, «pr» para «print»), aunque a veces esto no se cumple, con lo que resulta difícil recordarlos todos.

— Las opciones permiten indicar las variantes del comando considerado. Estas opciones están normalmente identificadas con un carácter precedido o no del signo « – ». Se pueden dar varias opciones para el mismo mandato.

— Los argumentos son cadenas de caracteres que a menudo representan nombres de ficheros sobre los cuales se aplica el mandato.

Para indicar la no obligatoriedad de las opciones o argumentos se encierran entre corchetes «[]», con lo que normalmente la sintaxis de un comando puede venir expresada de la siguiente manera:

nombre   [opción]   [fichero]

Después de un comando SIEMPRE hay que pulsar la tecla RETURN. Esto es algo sobre lo que no se insistirá más a lo largo de este libro.



MULTIPLES ORDENES EN UNA MISMA LINEA

Se pueden teclear dos o más órdenes en una misma línea si están separadas por puntos y comas (;). El punto y coma le indica al shell que ejecute la primera orden, que espere a que se acabe y ejecute luego la orden siguiente. Por ejemplo, teclee **«date;who»:**

```
%date;who
Tue Dec 23 17:40:00 GMT 1986
f1457     tty02    Dec 23 17:24
e0254     tty03    Dec 23 17:22
luisd     console Dec 23 17:28
%
```

En la pantalla aparece la fecha y la hora del día (ejecución del comando **date**), y luego una lista de todos los usuarios que están conectados actualmente al sistema (ejecución del comando **who**).



REDIRECCION DE LA ENTRADA Y LA SALIDA



Redirección de la salida a un fichero

El sistema operativo UNIX envía la ejecución de un programa o de un comando a un fichero llamado **fichero estándar de salida.** Normalmente el fichero estándar de salida se asigna a la pantalla del terminal. El shell nos permite «dirigir» o «redireccionar» la salida estándar a otro fichero especificado en la línea de órdenes.

orden > fichero

El signo «>» indica que «fichero» es la salida para «orden», es decir, que el resultado de la ejecución de ese mandato ya no se verá por pantalla, sino que se quedará almacenado en «fichero». Si éste no existe, será creado por el shell. Si existe, se destruye todo su contenido anterior y pasará a almacenar la salida de «orden».

Teclee por ejemplo **date > mifichero».** Ahora compruebe con el comando *cat* (que se verá más adelante) el contenido de «mifichero», y verá cómo tiene almacenada la fecha y la hora del día.

```
%date > mifichero
%cat mifichero
Tue Dec 23 17:42:37 GMT 1986
%
```

Cuando necesite redirigir la salida de dos o más órdenes a un fichero, encierre éstas entre paréntesis. Por ejemplo, teclee **«(date;who) > mifichero».** Ahora compruebe con el comando *cat* el resultado.

```
%(date;who)>mifichero
%cat mifichero
Tue Dec 23 17:44:37 GMT 1986
f1457     tty02    Dec 23 17:24
e0254     tty03    Dec 23 17:22
luisd     console Dec 23 17:28
%
```

Si no se hubieran encerrado entre paréntesis las órdenes **date** y **who,** el shell hubiera ejecutado **date,** mostrando el resultado en el terminal, y luego hubiera ejecutado **who,** guardando el resultado en el fichero «mifichero».

A veces puede interesar añadir la salida de un proceso a un fichero sin borrar lo que hay en él. Para ello se dispone del operador «>>». Teclee **«date** >> mifichero». Ahora compruebe con el comando **cat** el fichero «mifichero». Verá que no se ha borrado el contenido anterior de «mifichero».

```
%date >> mifichero
%cat mifichero
Tue Dec 23 17:44:37 GMT 1986
f1457     tty02    Dec 23 17:24
e0254     tty03    Dec 23 17:22
luisd     console Dec 23 17:28
Tue Dec 23 17:45:46 GMT 1986
%
```



Redirección de la entrada

El sistema operativo UNIX también tiene definido un fichero estándar de entrada, el cual se asigna al teclado. El shell permite redireccionar este fichero estándar de entrada a cualquier otro fichero en disco, normalmente creado mediante un editor de texto, con lo cual la lectura de datos ya no se efectúa del teclado, sino de ese fichero.

La redirección se realiza con el signo «<»:

orden < fichero

De manera general, si queremos redireccionar la entrada y la salida, la sintaxis de un comando será:

orden < FicheroDeEntrada > FicheroDeSalida



CONEXION DE COMANDOS MEDIANTE PIPE-LINES

El sistema operativo UNIX permite conectar programas (los comandos también son programas), de manera que cada uno trabaja con unos datos proporcionados como entrada, los transforma y produce unos determinados datos como salida. Estos datos de salida pueden ser tomados, a su vez, como datos de entrada para otro programa, y así sucesivamente.



Fig. 3.1. Esquema de una conexión entre 3 programas distintos.

La conexión se realiza a través de «tubos» **(pipelines).** El operador utilizado para esto es el símbolo « | », y el formato de dos órdenes conectadas por un pipe es la siguiente:

orden1 | orden2

La salida de «orden1» no aparece en el terminal ni se almacena en un fichero intermedio. Simplemente, la salida del primer proceso sirve de entrada al segundo (a «orden2»).



Fig. 3.2. Conexión de programas mediante pipes.

Por ejemplo, teclee **«who | sort».**

```
%who
f1457    tty02    Dec 23 17:24
e0254    tty03    Dec 23 17:22
luisd    console  Dec 23 17:28
%who | sort
e0254    tty03    Dec 23 17:22
f1457    tty02    Dec 23 17:24
luisd    console  Dec 23 17:28
%
```

El resultado es que los usuarios salen ordenados en pantalla. Ello es debido a que la salida de la orden **who** (dice quién está conectado al sistema) sirve como entrada a la orden **sort,** que ordena alfabéticamente los datos que recibe como entrada (que en este caso es la salida del comando anterior).



CARACTERES ESPECIALES Y DE SUSTITUCION

Existe un cierto número de caracteres que tienen un significado especial para el shell. Son los siguientes:

* ? | < > ; : [ ] &

Estos caracteres no se deberán utilizar en nombres de fichero debido a su interpretación especial. Algunos de ellos ya se han visto anteriormente y se ha explicado su utilidad. Los que interesan ahora son los llamados caracteres de sustitución, que son el asterisco («*»), la interrogación («?») y los corchetes («[ ]»). El significado de los mismos es:

* Cualquier cadena de caracteres (excepto /), incluida la cadena vacía.
? Cualquier carácter sencillo.

[...] Cualquier carácter que pertenezca a la lista dada entre corchetes. Dos caracteres separados por un guión «-» definen una lista de caracteres clasificados según el orden léxico.

Por ejemplo:

% orden * «Orden» actúa sobre cualquier fichero del directorio actual. Este último concepto se verá en el capítulo 5.

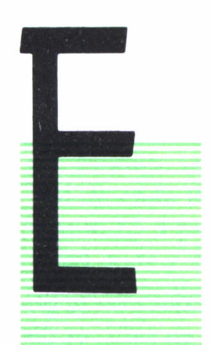
% orden *.tex «Orden» actúa sobre todos los ficheros cuya extensión es «.tex».

% orden prog.* «Orden» actúa sobre todos los ficheros que se llamen «prog» y cuya extensión sea cualquiera.

% orden cap? «Orden» actúa sobre todos los ficheros cuyo nombre empieza por «cap», seguido de algún carácter sencillo.

% orden practica [1-3] «Orden» actúa sobre los ficheros «practica1», «practica2» y «practica3».

COMUNICACION ENTRE USUARIOS 4



EL sistema operativo UNIX permite intercambiar mensajes entre los usuarios. Para ello dispone de dos modalidades:

— Intercambio de mensajes por medio de «buzones de cartas», a través de la orden **mail.**
— Intercambio de mensajes entre usuarios de manera interactiva, mediante la orden **write.**



COMUNICACION MEDIANTE BUZONES DE CARTAS



Recepción del correo

Una de las muchas funciones del programa **login,** aparte de permitirle entrar en el sistema, es la de advertir de la existencia de correo en el buzón de cartas, mediante la aparición en pantalla de la frase «You have mail».

```
:login: domingo
Password:

                              Bienvenidos a Unix !!

You have mail.

%
```

Si al conectarse se encuentra con que hay correo, la manera de visualizar los mensajes que nos han enviado otros usuarios es mediante el comando **mail.**

```
%mail
From e0254 Tue Dec 23 17:22:30 1986
recuerda eso que te dije

? From f1457 Tue Dec 23 17:22:19 1986
hoy a las 18 h. borro los ficheros

? From f1457 Tue Dec 23 17:21:46 1986
hoy tenemos una reunion a las 10 h.
```

La sintaxis del mandato **mail** es:

mail [-r] [-q] [-p] [-f NombreDeFichero]

. Si se teclea la orden sin ninguna opción, salen los mensajes con la política de que el último en entrar (en el buzón) es el primero en salir, es decir, el mensaje más reciente es el primero que sale en la pantalla.

Las opciones significan lo siguiente:

-r Invierte el orden en que mail presenta los mensajes, y ahora salen con política «primero en entrar, primero en salir».

-q Se sale de mail después de que se teclee un carácter de interrupción.

-p Saca por pantalla todos los mensajes del buzón sin hacer preguntas, y luego sale de la orden mail.

-f NombreDeFichero Muestra en la pantalla el fichero nombrado como si fuese el buzón de cartas.

El programa **mail,** a menos que se utilice con la opción **«-p»,** mostrará su propio prompt (el carácter de interrogación (?)), y esperará a que se le den indicaciones para tratar el mensaje. Estas indicaciones pueden ser las siguientes:

RETURN Visualiza el mensaje siguiente o sale del programa mail en caso de que sea el último.

d Destruye el último mensaje aparecido en pantalla y visualiza el siguiente.

p Imprime el mensaje de nuevo.

- Muestra otra vez el mensaje anterior.

s[fichero] Guarda los mensajes en el fichero indicado. Si no se le espe-

cifica un nombre de fichero se guarda en el fichero estándar **mbox.** A continuación se sale del programa mail y vuelve a aparecer el promt del sistema.

m[usuario] Envía el mensaje al (los) usuario(s) especificado(s). Por defecto, usted es el usuario.

q o **ctrl+d** Deja sin visualizar los mensajes que queden en el buzón y sale del programa mail.



Envío de correo

La orden **mail** también sirve para enviar mensajes a otros usuarios. Para ello después de la palabra mail se deberá poner el nombre de conexión del usuario al que se quiera mandar el mensaje.

Cuando se está construyendo el mensaje se debe teclear RETURN al final de cada línea. Se puede terminar con una línea que sólo contenga un punto (.) o pulsando **«ctrl + d»**, pero ya que es fácil que se tecleen varios «ctrl + d» y desconectarse del sistema, lo mejor es tomar como regla general el acabar los mensajes con un punto.

```
%mail domingo
Esto es un ejemplo de mandar mensajes entre usuarios
.
%
```

Con la orden mail también se puede mandar mensajes uno mismo, de manera que le sirvan de recordatorio para la próxima vez que se conecte. Para ello deberá poner su nombre de conexión después de la palabra mail.



COMUNICACION INTERACTIVA ENTRE USUARIOS

Mediante la orden **write** puede realizarse un intercambio conversacional de mensajes entre dos usuarios que estén conectados al sistema. Para ello, ambos deben ejecutar un comando **write** destinado al otro.

La sintaxis de este mandato es la siguiente:

write NombreDeConexión [identificador del terminal]

Es conveniente utilizar primero la orden **who** para ver si el receptor deseado está conectado al sistema. Además, ya que un usuario puede estar conectado a más de un terminal a la vez, puede que haya que especificar el identificador del terminal del receptor.

Cuando el emisor ejecuta el comando **write**, en la pantalla del receptor aparece el aviso:

Message from SuNombreDeConexión Ident.delTerminal

lo cual le indica que de un momento a otro recibirá un mensaje de un emisor, cuyo identificador de entrada y terminal en el que está trabajando se especifican.

Los mensajes se envían cada vez que se pulsa un retorno de carro (la tecla RETURN). Para evitar que éstos se crucen y se confundan en pantalla se recomienda utilizar el siguiente protocolo:

— Cuando se escribe por primera vez a otro usuario, espere a que éste responda antes de mandar su mensaje.

— Cada interlocutor terminará cada mensaje con una señal distintiva, por ejemplo - o - (over/terminado).

— Cuando se desee terminar la conversación, el mensaje finalizará con oo (over and out), y después de la recepción de la misma intención por parte del otro interlocutor se terminará el enlace pulsando **«ctrl + d»** (final de fichero).

Un ejemplo de conversación sería el siguiente:

— pantalla del usuario — con nombre de conexión domingo	— pantalla del usuario — con nombre de conexión pedro
% write pedro	%
	Message from domingo tty01
	% write domingo
Message from pedro tty02	
¿Cuándo vas a darme aquello que te dije?	
o	
	¿Cuándo vas a darme aquello que te dije?
	o
	Mañana a las 5
	o
Mañana a las 5	
o	
Vale, de acuerdo	
oo	

— pantalla del usuario — con nombre de conexión domingo	— pantalla del usuario — con nombre de conexión pedro
	Vale, de acuerdo
	oo
	Hasta entonces
	oo
Hasta entonces	
oo	
«ctrl + d»	«ctrl + d»
%	%



PERMISO O PROHIBICION DE MENSAJES

Con la orden **mesg** se puede permitir o prohibir el acceso a su terminal para la orden **write,** de manera que si se tiene el permiso denegado, aunque otro usuario intente comunicar con usted mediante **write,** usted no será molestado y los mensajes de ese otro usuario no aparecerán en la pantalla.

La sintaxis de esta orden es:

mesg [n] [y]

Las opciones significan lo siguiente:

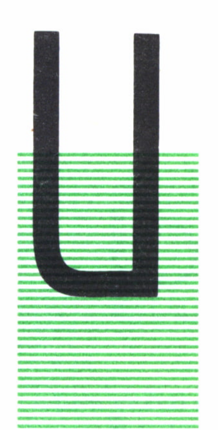
mesg n	Quita el permiso de escritura, es decir, nadie le molestará.
mesg y	Repone el permiso de escritura.
mesg	Proporciona el estado actual

```
%mesg
is y
%
```

Si mediante **write** se intenta comunicar con alguien que tiene su permiso de escritura denegado, aparecerá un mensaje como el siguiente:

```
%write to ttv02
write: permission denied
%
```

ADMINISTRACION DE FICHEROS 5



SISTEMA DE FICHEROS. CONCEPTO DE DIRECTORIO



N directorio es un fichero que contiene un conjunto de líneas. Cada línea es el nombre de un fichero o un subdirectorio.

El sistema de ficheros es la estructura en que un sistema operativo tiene organizados los ficheros. En UNIX esta estructura está dispuesta según una jerarquía de directorios en forma de árbol invertido, donde los nodos son los directorios y las hojas son los ficheros normales. La fuente del árbol es la raíz o el directorio raíz, señalado por el carácter «/». Cada directorio puede contener cualquier número de subdirectorios, los cuales, a su vez, tendrán otros subdirectorios y/o ficheros, desarrollándose así una estructura ramificada.



NOMBRE DE CAMINO DE UN FICHERO

Un nombre de camino especifica el nombre completo de un directorio o un fichero, empezando desde la raíz y siguiendo la jerarquía de directorios en forma de árbol invertido. Los distintos nombres que componen el camino de acceso están separados por el carácter «/». En la figura 5.1 un ejemplo de nombre de camino completo para el fichero «carta.tex» sería:

/usr/domingo/texto/carta.tex

Cada fichero y cada directorio tienen un nombre de camino único que identifica ese fichero, pero este método para designarlos puede ser muy lento cuando el número de directorios se incrementa considerablemente, ya que entonces el camino de acceso absoluto se compone de muchos nombres. Por ello es por lo que un fichero también puede referenciarse por un nombre relativo. Tomando el ejemplo anterior, si se estuviera en

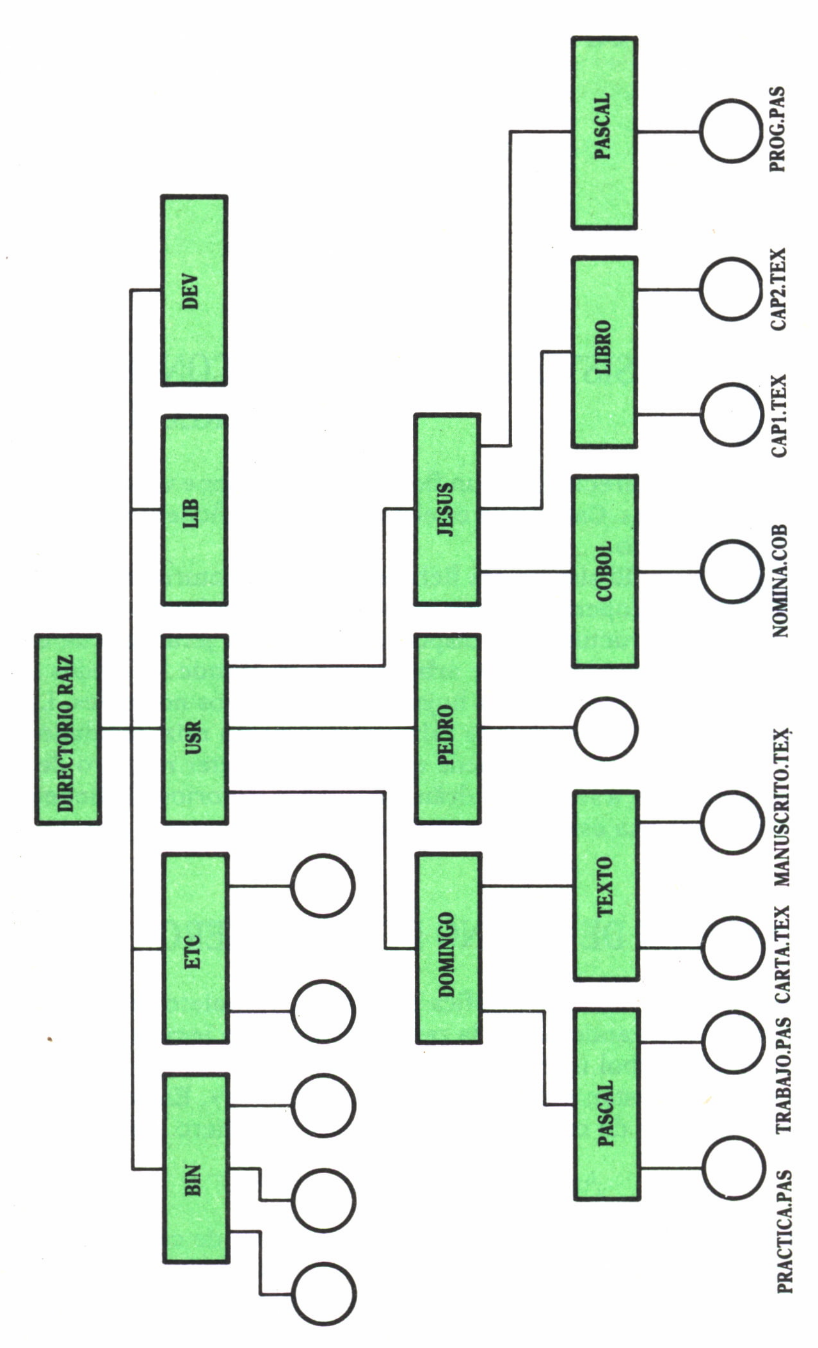


Fig. 5.1. Esquema del sistema de ficheros de UNIX (los rectángulos representan directorios y los círculos ficheros de datos).

el directorio «domingo», el fichero «carta.tex» se referenciaría por «texto/carta.tex», y si se estuviera en el directorio «texto», se referenciaría sólo por «carta.tex».

En general, a todo usuario se le asocia un **directorio actual** por defecto. A partir de éste el usuario construye su propio sub-árbol de directorios y ficheros. Al comienzo de cada sesión el sistema asocia el directorio del usuario como directorio actual de todos sus procesos. En el ejemplo anterior el usuario con nombre de conexión «domingo» al entrar en el sistema ya estaría situado en el directorio /usr/domingo.

Si en cualquier momento de la sesión necesita saber en qué directorio de trabajo está, utilice el comando **pwd,** que muestra en pantalla el nombre de camino completo, empezando en la raíz y siguiendo a través de cualquier directorio intermedio hasta encontrar su directorio actual.

```
%pwd
/usr1/alumnos/tercero/e0254/libro
%
```

Esta orden es bastante útil si se olvida en qué punto del sistema de ficheros se halla. Sólo tiene acceso inmediato a cualquier fichero o directorio del directorio actual, a menos que se especifiquen los nombres de camino completos.



VISUALIZACION DEL CONTENIDO DE UN DIRECTORIO

Con el comando **ls** se puede «listar» el contenido de un directorio, es decir, se visualizan ordenados alfabéticamente los nombres de todos los ficheros y subdirectorios del directorio especificado, proporcionando opcionalmente información sobre permisos de lectura/escritura, fecha del último cambio, etc.

La sintaxis de esta orden es:

ls **[opciones]** **[NombreDeDirectorio]**

Si no se especifica nigún directorio, el sistema toma por defecto el directorio actual.

```
%ls
carta.tex
documento.tex
nomina.cob
programa.c
prueba.pas
%
```

En el caso de un fichero normal, el comando **ls** sólo proporciona las informaciones concernientes al fichero nombrado. En el caso de un nombre de directorio, proporciona todas las informaciones solicitadas mediante las opciones sobre todos los ficheros del directorio.

Algunas opciones de este comando son:

-l Lista en formato «largo», es decir, proporciona el conjunto de características de los ficheros: tipo, protección, número de enlaces, propietario, tamaño, fecha de la última modificación y nombre.

```
%ls -l
total 41
-rw-r--r--   1 e0254    2374      3626 Dec 30 13:48 carta.tex
-rwxr-xr-x   1 e0254    2374      8636 Dec 30 13:50 documento.tex
-rw-r--r--   1 e0254    2374      4564 Dec 30 13:47 nomina.cob
-rw-r--r--   1 e0254    2374      2085 Dec 30 13:44 programa.c
-rw-r--r--   1 e0254    2374       389 Dec 30 13:41 prueba.pas
%
```

Los diez primeros caracteres que aparecen con esta opción se interpretan de la siguiente manera: si el primer carácter es:

d el fichero es un directorio
el fichero es un fichero ordinario.

Los nueve caracteres siguientes se interpretan como tres conjuntos de tres caracteres cada uno. Su significado se explicará en profundidad cuando se trate la protección de ficheros. Como adelanto, decir que el primer conjunto se refiere a los permisos del propietario, el segundo a los permisos de los usuarios del mismo grupo que el propietario, y el tercero a los permisos de todos los demás usuarios. Dentro de cada conjunto, cada carácter indica, respectivamente, permisos para leer, modificar (escribir) y ejecutar el fichero (como un programa). Para un directorio el permiso de ejecutar significa que se puede buscar en el directorio un fichero específico. Estos tres caracteres a los que se hace referencia son:

r El fichero se puede leer.
w El fichero se puede modificar (escribir).
x El fichero se puede ejecutar.
- No se concede el permiso indicado.

-t Lista ordenando según la última modificación (primero la más reciente).

```
%ls -t
documento.tex
carta.tex
nomina.cob
programa.c
prueba.pas
%
```

-s Informa del tamaño en bloques.
(1 bloque = 512 ó 1024 caracteres, según las versiones UNIX.)

```
%ls -s
total 41
    8 carta.tex
   18 documento.tex
    9 nomina.cob
    5 programa.c
    1 prueba.pas
%
```

«Total» indica el total de bloques, y a cada fichero le precede el número de bloques que tiene.

-r El contenido del directorio aparece ordenado alfabéticamente en orden inverso.

```
%ls -r
prueba.pas
programa.c
nomina.cob
documento.tex
carta.tex
%
```

-u Lista ordenando según el último acceso (primero el más reciente).



CREACION DE UN DIRECTORIO

Para crear un nuevo directorio es preciso disponer del permiso de escritura sobre el directorio al que se va a incorporar. De ser así, se puede crear mediante el comando **mkdir.**

La sintaxis del mismo es:

mkdir **NombreDeDirectorio(s)**

Puede crear uno o varios a la vez, según se especifique. Por defecto, los directorios se crean con los permisos de consulta (r) y ejecución (x) para todos los usuarios. Para la modificación (w) sólo está autorizado el usuario.

```
%ls -l
total 41
-rw-r--r--   1 e0254    2374        3626 Dec 30 13:48 carta.tex
-rwxr-xr-x   1 e0254    2374        8636 Dec 30 13:50 documento.tex
-rw-r--r--   1 e0254    2374        4564 Dec 30 13:47 nomina.cob
-rw-r--r--   1 e0254    2374        2085 Dec 30 13:44 programa.c
-rw-r--r--   1 e0254    2374         389 Dec 30 13:41 prueba.pas
%mkdir pascal
%ls -l
total 42
-rw-r--r--   1 e0254    2374        3626 Dec 30 13:48 carta.tex
-rwxr-xr-x   1 e0254    2374        8636 Dec 30 13:50 documento.tex
-rw-r--r--   1 e0254    2374        4564 Dec 30 13:47 nomina.cob
drwxr-xr-x   2 e0254    2374          32 Dec 30 14:01 pascal
-rw-r--r--   1 e0254    2374        2085 Dec 30 13:44 programa.c
-rw-r--r--   1 e0254    2374         389 Dec 30 13:41 prueba.pas
%
```



CAMBIO DE DIRECTORIO

Con la orden **cd** se cambia el directorio de trabajo actual. Su sintaxis es:

cd
cd NombreDeDirectorio

Con **cd** se vuelve al directorio de conexión desde cualquier directorio. Con **cd Subdirectorio** se traslada desde el directorio actual hasta el subdirectorio nombrado. Se supone que el mismo ya ha sido creado mediante el comando **mkdir.**

Con **cd NombreDeRecorridoCompleto** se traslada al último directorio del nombre de recorrido dado, por ejemplo, con:

cd /usr/alumnos/tercero/e0254

se trasladaría al directorio «e0254».

Una variante muy útil de esta orden es la que sube un nivel en la jerarquía de directorios, es decir, cambia al directorio «padre». Se consigue con:

cd ..

De manera análoga, para cambiar al directorio «abuelo» (es decir, al «padre» del directorio «padre») se teclea:

cd ../..

y así sucesivamente.

```
% cd direcpruebas
% pwd
/usr1/alumnos/tercero/e0254/libro/direcpruebas
%
```

CREACION DE UN FICHERO

Para crear un fichero hay que disponer del derecho de escritura (w) en el directorio en el cual se va a incorporar ese fichero.

La creación de un fichero se realiza normalmente mediante la utilización de un editor de texto (ed o vi) o de un procesador de textos (nroff). El editor vi se verá en profundidad en un capítulo posterior.

También puede crearse un fichero de manera automática, como, por ejemplo, los ficheros creados por un compilador, o también utilizando un redireccionamiento del fichero estándar de salida.

VISUALIZACION DEL CONTENIDO DE UN FICHERO

El contenido de un fichero se puede visualizar mediante el comando **cat.** Cat es la abreviatura de concatenar, que significa encadenar juntos; por tanto, cuando se desee mostrar en pantalla más de un fichero, el contenido de éstos se encadenará, apareciendo uno detrás de otro.

También puede mostrarse por pantalla el contenido de un fichero me-

diante un editor de texto, pero si está en el shell la utilización de **cat** es más directa.

La sintaxis de esta orden es:

cat NombreDeFichero(s)

```
%who>quien
%cat quien
e0254     tty02    Dec 30 13:32
e0411     tty04    Dec 30 11:46
sec1      console Dec 30 09:36
%
```

Un ejemplo de encadenamiento sería:

```
%date>fecha
%cat quien fecha
e0254     tty02    Dec 30 13:32
e0411     tty04    Dec 30 11:46
sec1      console Dec 30 09:36
Tue Dec 30 14:13:09 GMT 1986
%
```

También puede redireccionarse la salida de este comando:

```
%cat quien fecha > ambos
%cat ambos
e0254     tty02    Dec 30 13:32
e0411     tty04    Dec 30 11:46
sec1      console Dec 30 09:36
Tue Dec 30 14:13:09 GMT 1986
%
```

De igual manera, puede hacerse que la orden **cat** acepte las líneas siguientes directamente desde el teclado y las escriba en un fichero. Se termina **cat** pulsando «ctrl + d». Para hacer las correcciones mientras se teclean líneas de texto se puede utilizar «#» para borrar el último carácter tecleado, y « @ » para borrar la última línea.

```
%cat > ficheroprueba
Esto es una prueba del comando cat.
Lo que se teclea se guarda en un fichero directamente.
((ctrl+d))
%cat ficheroprueba
Esto es una prueba del comando cat.
Lo que se teclea se guarda en un fichero directamente.
%
```

En caso de que se especifique un fichero que no exista o no pueda leerse, la orden **cat** dará un mensaje de error como el siguiente:

```
%cat otrofichero
cat: cannot open otrofichero
%
```



ELIMINACION DE FICHEROS Y DIRECTORIOS

Mediante la orden **rm** pueden eliminarse uno o más nombres de ficheros. Su sintaxis es:

rm [opciones] nombre(s)

Este comando elimina un fichero desuniendo y borrando su referencia del directorio. Si el nombre pasado como argumento constituía la única referencia a ese fichero, éste también se destruye.

La eliminación de un fichero requiere permiso para escribir en el directorio del cual está «enganchado». Si se trata de eliminar un fichero que no tiene permiso de escritura, un mensaje indicará que el fichero es sólo de lectura.

Las opciones del mandato **rm** son las siguientes:

-f Fuerza la eliminación de ficheros sin permiso de escritura.

-r Borra el contenido total de un directorio, así como el directorio mismo.

-i Solicita del operador una confirmación para borrar cada fichero.

Algunos ejemplos de utilización de este comando son:

```
%ls                          %rm ambos  ficheroprueba
ambos                        %ls
carta.tex                    carta.tex
documento.tex                documento.tex
fecha                        fecha
ficheroprueba                nomina.cob
nomina.cob                   pascal
pascal                       programa.c
programa.c                   prueba.pas
prueba.pas                   quien
quien                        %
```

En el ejemplo siguiente se borran los ficheros especificados después de pedir confirmación y sólo en caso de que ésta sea positiva (y).

```
%rm -i *.*
carta.tex: y
documento.tex: n
nomina.cob: n
programa.c: n
prueba.pas: y
%ls
documento.tex
fecha
nomina.cob
pascal
programa.c
quien
%
```

También pueden concatenarse opciones, como muestra el siguiente ejemplo, que elimina todos los ficheros (y directorios si están vacíos) del directorio «pascal», después de pedir la confirmación.

```
%rm -ri pascal
directory pascal: y
```

```
pascal: y
%ls
documento.tex
fecha
nomina.cob
programa.c
quien
%
```

Para borrar un directorio se utiliza la orden **rmdir.** En este caso el directorio tiene que estar vacío; si no, aparecerá un mensaje de error, como muestra el siguiente ejemplo:

```
%rmdir libro
rmdir: libro not empty
%
```



BUSQUEDA DE UN FICHERO

Mediante la orden **find** pueden buscarse ficheros en una arborescencia del sistema de ficheros, así como la ejecución de un mandato para cada uno de los ficheros encontrados. La sintaxis de este comando es:

find NombreDeFichero(s) condiciones

Este mandato recorre la arborescencia de ficheros para encontrar todos los que tengan las condiciones especificadas, las cuales pueden ser las siguientes:

-name NombreDeFichero	Busca todos los ficheros que concuerden con NombreDeFichero.
-type T	Busca todos los ficheros cuyo tipo sea T, donde T es «f» para fichero normal y «d» para directorio.
-user NombreDeUsuario	Busca los ficheros con el nombre de propietario «NombreDeUsuario».

-group NombreDeGrupo	Busca los ficheros cuyo nombre de grupo es el especificado.
-size n	Localiza los ficheros cuyo tamaño sea de «n» bloques.
-atime n	Busca los ficheros a los que se ha accedido hace «n» días.
-mtime n	Localiza los ficheros modificados hace «n» días.
-exec mandato {}	Ejecuta el mandato UNIX especificado, para cada fichero encontrado que cumpla las condiciones expuestas en la orden. «Mandato» que debe terminar por «;», y las llaves «{}» representan un argumento para la orden.
-ok mandato	Igual que la condición anterior, pero antes de ejecutar el mandato pide al usuario confirmación. Se teclea una «y» para «sí», y «n» en caso contrario.
-print	Muestra en pantalla el nombre de recorrido de los ficheros que cumplen las condiciones especificadas.

Para las condiciones existen los siguientes argumentos y convenios:

n	Entero decimal que significa «n exactamente».
-n	Menor que «n».
+n	Mayor que «n».
.	La búsqueda se inicia en el directorio de conexión.
()	Se utilizan paréntesis para agrupar las condiciones que deban ser satisfechas simultáneamente por los ficheros que vayan encontrando.
-a	Si se especifica más de una condición, se buscan los ficheros que las cumplan simultáneamente (condición lógica AND).
-o	Precede a un conjunto de condiciones de búsqueda de ficheros. Se buscan los ficheros que cumplan cualquiera de las condiciones especificadas (operador lógico OR).
-	Todas las condiciones empiezan con el carácter «-».

Algunos ejemplos del comando **find** son los siguientes:

```
% find . -name "*.pas" -print
direcpruebas/programa.pas
direcpruebas/practica.pas
%
```

Busca a partir del directorio de conexión («.») todos los ficheros que tengan como extensión «.pas» y los imprime.

```
% find . -name "*.c" -size +1 -print
direcpruebas/programa.c
%
```

Busca a partir del directorio de conexión todos los ficheros con extensión «.c» que ocupen más de un bloque e imprime sus nombres en el fichero estándar de salida (normalmente la pantalla del terminal).

```
% find . -name "*" -print
direcpruebas
direcpruebas/programa.pas
direcpruebas/programa.c
direcpruebas/nomina.cob
direcpruebas/practica.pas
direcpruebas/documento.tex
direcpruebas/nucleo.c
direcpruebas/quien
direcpruebas/fecha
%
```

Busca a partir del directorio de conexión todos los nombres posibles de ficheros e imprime sus nombres. Cuando se desee encontrar todos los ficheros, se puede omitir la opción «-name», ya que el caso por defecto especifica todos los ficheros. El ejemplo siguiente obtiene lo mismo que el anterior:

```
% find . -print
direcpruebas
direcpruebas/programa.pas
direcpruebas/programa.c
direcpruebas/nomina.cob
direcpruebas/practica.pas
direcpruebas/documento.tex
direcpruebas/nucleo.c
direcpruebas/quien
direcpruebas/fecha
%
```

En el ejemplo siguiente se busca a partir del directorio de conexión todos los ficheros que no se han modificado en los últimos dos días e imprime sus nombres.

```
% find . -mtime +2 -print
direcpruebas/programa.c
direcpruebas/nomina.cob
direcpruebas/documento.tex
direcpruebas/quien
direcpruebas/fecha
%
```

Busca a partir del directorio de conexión todos los ficheros que tengan la extensión «.c» o «.pas» que ocupen más de dos bloques y menos de diez, e imprime sus nombres.

```
% find . \( -name "*.c" -o -name "*.pas" \) -size +2 -size -10 -print
direcpruebas/programa.c
direcpruebas/practica.pas
%
```

Obsérvese que pueden concatenarse más condiciones agrupadas en el sentido lógico AND añadiéndolas a la línea de órdenes. Otra cosa que hay que resaltar es que para que el shell interprete correctamente los paréntesis, deben estar precedidos por el carácter «\».

El ejemplo siguiente busca a partir del directorio de conexión todos los ficheros con la extensión «.c», y pide su consentimiento (-ok) para ejecutar la orden **rm** (borrar los ficheros).

```
% find . -name "*.c" -ok rm \;
< rm ... direcpruebas/programa.c >?    n
< rm ... direcpruebas/nucleo.c >?    y
%
```

PROTECCION DE FICHEROS

El sistema operativo UNIX distingue para cada fichero tres niveles de derecho de acceso:

— el propietario
— el grupo de su propietario
— el resto de los usuarios

Para cada uno de estos niveles el propietario de un fichero puede asignar tres tipos de autorizaciones:

— de lectura
— de escritura
— de ejecución

El permiso de lectura garantiza el acceso para imprimir el contenido de un fichero, o para listar un directorio. El de escritura permite añadir caracteres a un fichero o añadir entradas («colgar» ficheros) a un directorio. El permiso de ejecución tiene un significado diferente, según sean ficheros o directorios. Si un fichero tiene el permiso de ejecución, se le puede llamar como si fuera una orden. Si un directorio tiene el permiso de ejecución, se puede acceder a sus nombres de ficheros, o buscar a través de él.

Los privilegios de acceso son independientes unos de otros. La presencia o ausencia de algún modo de acceso no afecta a ningún otro modo.

El sistema operativo UNIX proporciona una orden para cambiar los modos permitidos de uno o más ficheros. La sintaxis de este mandato es:

chmod [quien] operador permiso fichero

El significado de cada argumento es:

Quién:

u Propietario (user/usuario).

g Grupo (group).

o Todos los demás (others).

a Todos: usuario, grupo y los demás.

Operador:

+ Añadir permiso.

- Quitar permiso.

= Asignar permiso absoluto para el fichero.

Permiso:

r Leer (read).

w Escribir (write).

x Ejecutar (execute).

Es preciso señalar que entre **«quién operador permiso»** no tiene que haber ningún espacio en blanco, pues provocaría un error.

En los ejemplos siguientes sobre la utilización de **chmod** se ha incluido el listado en formato «largo» de los ficheros utilizados, antes y después de la ejecución del comando, de manera que se vean claros sus efectos.

```
% ls -l documento.tex
-rwxrwxrwx   1 e0254    2374         8636 Dec 30 13:50 documento.tex
% chmod o-w documento.tex
% ls -l documento.tex
-rwxrwxr-x   1 e0254    2374         8636 Dec 30 13:50 documento.tex
%
```

Se quita el permiso de escritura sobre el fichero especificado a los usuarios que no son ni el propietario ni los de su grupo (a los «otros»).

```
% ls -l nomina.cob
-rw-rw-rw-   1 e0254    2374         4564 Dec 30 13:47 nomina.cob
% chmod go-w nomina.cob
% ls -l nomina.cob
-rw-r--r--   1 e0254    2374         4564 Dec 30 13:47 nomina.cob
%
```

Se quita el permiso de escritura sobre el fichero especificado a los usuarios del mismo grupo que el propietario y a todos los demás.

```
% ls -l nomina.cob
-rw-r--r--   1 e0254    2374         4564 Dec 30 13:47 nomina.cob
% chmod a=rwx nomina.cob
% ls -l nomina.cob
-rwxrwxrwx   1 e0254    2374         4564 Dec 30 13:47 nomina.cob
%
```

Se cambia el modo para todos los usuarios asignando permiso para leer, escribir y ejecutar el fichero nombrado.

También se puede con el comando **chmod** cambiar el modo permitido para uno o varios directorios especificados. La sintaxis es:

chomd [quien] operador permiso directorio

```
% ls -l
total 41
drwxr--r--    2 e0254    2374          192 Jan  5 18:15 direcpruebas
-rw-r--r--    1 e0254    2374           91 Dec 30 15:22 fig_2.1
-rw-r--r--    1 e0254    2374           49 Dec 23 18:00 fig_2.2
-rw-r--r--    1 e0254    2374          222 Dec 30 14:27 list_5.11
% chmod -x direcpruebas
% ls -l
total 46
drw-r--r--    2 e0254    2374          192 Jan  5 18:15 direcpruebas
-rw-r--r--    1 e0254    2374           91 Dec 30 15:22 fig_2.1
-rw-r--r--    1 e0254    2374           49 Dec 23 18:00 fig_2.2
-rw-r--r--    1 e0254    2374          222 Dec 30 14:27 list_5.11
%
```

Cambia de modo para todos los usuarios (opción por defecto), restringiendo el permiso de búsqueda en el directorio especificado.

Algunos mensajes de error del comando **chmod** son:

```
% chmod o + x programa.c
chmod: can't access +
chmod: can't access x
%
```

No se puede separar «quién» de «operador», y tampoco «operador» de «permiso».

```
% chmod o+x prueba
chmod: can't access prueba
%
```

No existe el fichero «prueba».



CAMBIO DE NOMBRES DE FICHEROS Y DIRECTORIOS

Con el comando **mv** pueden renombrarse ficheros o directorios, así como trasladar un grupo de ficheros a otro directorio. La sintaxis de esta orden es:

mv NombreAntiguo NuevoNombre

que cambia el nombre del fichero «NombreAntiguo» por el de «Nuevo-Nombre».

Otra forma de utilizar este comando es:

mv fichero(s) NombreDeDirectorio

que traslada uno o más ficheros al directorio indicado.

Esta orden no modifica en absoluto el contenido de los ficheros fuente, que pueden ser ficheros normales o directorios. Si el nuevo nombre que se le va a dar al fichero ya existe, el fichero que tenía ese nombre se destruye, a no ser que el usuario no tenga derecho de escritura, en cuyo caso el sistema interroga al usuario.

Si el fichero destinatario es un directorio y los ficheros fuente (los de partida) son ficheros normales, la orden los «engancha» del directorio indicado.

```
% ls -l
total 56
-rwxrwxr-x    1 e0254     2374          8636 Dec 30 13:50 documento.tex
-rw-r--r--    1 e0254     2374            29 Dec 30 14:13 fecha
-rw-r--r--    1 e0254     2374             8 Jan  7 14:40 list_5.23
-rwxrwxrwx    1 e0254     2374          4564 Dec 30 13:47 nomina.cob
-rwxrwxrwx    1 e0254     2374          4564 Jan  5 16:18 practica.pas
-rw-r--r--    1 e0254     2374          2085 Dec 30 13:44 programa.c
-rw-r--r--    1 e0254     2374          5632 Jan  5 16:18 programa.pas
-rw-r--r--    1 e0254     2374            90 Dec 30 14:08 quien
% mv programa.pas otro.pas
% ls -l
total 57
-rwxrwxr-x    1 e0254     2374          8636 Dec 30 13:50 documento.tex
-rw-r--r--    1 e0254     2374            29 Dec 30 14:13 fecha
-rw-r--r--    1 e0254     2374           568 Jan  7 14:42 list_5.23
-rwxrwxrwx    1 e0254     2374          4564 Dec 30 13:47 nomina.cob
-rw-r--r--    1 e0254     2374          5632 Jan  5 16:18 otro.pas
-rwxrwxrwx    1 e0254     2374          4564 Jan  5 16:18 practica.pas
-rw-r--r--    1 e0254     2374          2085 Dec 30 13:44 programa.c
-rw-r--r--    1 e0254     2374            90 Dec 30 14:08 quien
%
```

Cambia el nombre del fichero «programa.pas» por el «otro.pas».

```
% ls -l nuevodirec
nuevodirec not found
% ls -l direcpruebas
total 55
-rwxrwxr-x    1 e0254     2374          8636 Dec 30 13:50 documento.tex
-rw-r--r--    1 e0254     2374            29 Dec 30 14:13 fecha
-rwxrwxrwx    1 e0254     2374          4564 Dec 30 13:47 nomina.cob
```

```
-rw-r--r--    1 e0254     2374          5632 Jan  5 16:18 otro.pas
-rwxrwxrwx    1 e0254     2374          4564 Jan  5 16:18 practica.pas
-rw-r--r--    1 e0254     2374          2085 Dec 30 13:44 programa.c
-rw-r--r--    1 e0254     2374            90 Dec 30 14:08 quien
% mv direcpruebas nuevodirec
% ls -l direcpruebas nuevodirec
direcpruebas not found
nuevodirec:
total 55
-rwxrwxr-x    1 e0254     2374          8636 Dec 30 13:50 documento.tex
-rw-r--r--    1 e0254     2374            29 Dec 30 14:13 fecha
-rwxrwxrwx    1 e0254     2374          4564 Dec 30 13:47 nomina.cob
-rw-r--r--    1 e0254     2374          5632 Jan  5 16:18 otro.pas
-rwxrwxrwx    1 e0254     2374          4564 Jan  5 16:18 practica.pas
-rw-r--r--    1 e0254     2374          2085 Dec 30 13:44 programa.c
-rw-r--r--    1 e0254     2374            90 Dec 30 14:08 quien
%
```

Cambia de nombre el directorio «direcpruebas» por el de «nuevodirec».



TAMAÑO DE UN FICHERO

Con la orden **wc** puede obtenerse el número de líneas, palabras y caracteres que tienen uno o más ficheros. La sintaxis de este mandato es:

wc [opcion(es)] NombreDeFichero(s)

donde las opciones son las siguientes:

-l Cuenta el número de líneas del fichero.
-w Cuenta el número de palabras del fichero.
-c Cuenta el número de caracteres.

```
% wc -l nomina.cob
     133 nomina.cob
%
```

El número que se obtiene es la cantidad de líneas que tiene el fichero indicado, seguido de su nombre. De manera análoga ocurre para las palabras y los caracteres.

```
% wc -w nomina.cob
     362 nomina.cob
% wc -c nomina.cob
    4564 nomina.cob
%
```

Cuando se especifica más de un fichero aparece al final una suma total.

```
% wc -l nomina.cob otro.pas
     133 nomina.qob
     156 otro.pas
     289 total
% wc -w nomina.cob otro.pas
     362 nomina.cob
     968 otro.pas
    1330 total
% wc -c noomina.cob otro.pas
    4564 nomina.cob
    5632 otro.pas
   10196 total
%
```

Las opciones se pueden concatenar. Por ejemplo:

```
% wc -lc nomina.cob
     133    4564 nomina.cob
%
```

La opción por defecto proporciona los tamaños en líneas, palabras y caracteres simultáneamente y por ese orden.

```
% wc nomina.cob
     133     362    4564 nomina.cob
% wc nomina.cob otro.pas
```

```
    133     362    4564 nomina.cob
    156     968    5632 otro.pas
    289    1330   10196 total
%
```

Si no se especifica ningún fichero, la orden **wc** cuenta la entrada del fichero estándar (el texto que se introduce por el teclado) hasta que se teclee un «ctrl + d».

```
% wc
esto es una prueba del comando wc
<CTRL+D>    1      7      34
%
```

Nota: En UNIX una palabra es una cadena de caracteres delimitada por espacios en blanco, tabuladores o caracteres de nueva línea («\n»).



INFORMACION DEL ESPACIO OCUPADO EN DISCO

Con la orden **du** se obtiene información detallada de la ocupación del disco en bloques de 512 bytes (1 carácter =1 byte).

La sintaxis de esta orden es:

du [opción] [NombreDeFichero]

Las opciones de este comando son dos, y no se pueden concatenar:

-s Sólo proporciona el número total de bloques para todos los ficheros.
-a Proporciona una indicación para cada fichero.

Si el «NombreDeFichero» es un directorio, se dan los bloques de disco que se han utilizado para éste y para todos los subdirectorios. Si no se especifica nada, el nombre por defecto es el directorio actual.

```
% du
57        .
% du -a
1         ./list_5.32
5         ./programa.c
```

```
9         ./nomina.cob
9         ./practica.pas
18        ./documento.tex
1         ./quien
1         ./fecha
12        ./otro.pas
57        .
% du -s
57        .
%
```

Esta orden puede ser útil en caso de que se necesite borrar algún fichero debido a que el espacio en disco se esté acabando, permitiendo así un uso más efectivo del mismo.



IMPRESION Y FORMATEO DE FICHEROS

Mediante el comando **lpr** se puede enviar un fichero a la cola de impresora para que se imprima de forma «desatendida». Su sintaxis es:

lpr [opciones] NombreDeFichero(s)

Las opciones son las siguientes:

-r Borra el fichero después de que se haya colocado en la cola de impresora.
-c Hace una copia del fichero para la cola de impresión y deja el original intacto (normalmente esta opción se toma por defecto).
-m Informa al usuario mediante un mensaje (mail) cuándo se ha terminado la impresión.
-n No informa por mail (normalmente opción por defecto).

Normalmente, antes de imprimir un fichero éste se formatea, es decir, se «retoca» un poco para que la salida obtenida se adapte a los formatos y necesidades del usuario. Esto se consigue con el comando **pr,** el cual permite separar las páginas de un fichero, imprimir cabeceras, añadir números de página, etc. La sintaxis de este comando es:

pr [opciones] [NombreDeFichero(s)]

Con la orden **pr fichero** se produce un listado de «fichero» preparado para que se imprima con la fecha y la hora, el nombre del fichero y el encabezamiento de página.

```
% pr quien

Dec 30 14:08 1986  quien Page 1

e0254     tty02    Dec 30 13:32
e0411     tty04    Dec 30 11:46
sec1      console  Dec 30 09:36

          .
          .      ( hasta la linea 66 en blanco )
          .
          .

%
```

Si no se especifica ningún nombre de fichero entonces se formatea la entrada estándar (lo que se introduzca por teclado).

Las opciones del comando **pr** son las siguientes:

-n Produce una salida en «n» columnas.
+n Empieza a imprimir el fichero en la página «n».
-h La cadena de caracteres que sigue entre comillas se toma como cabecera de página.
-wn Establece una anchura de página de «n» caracteres, y no de 72, que es la anchura tomada por defecto.
-ln Establece una longitud de página de «n» líneas, en vez de las 66 líneas tomadas por defecto.
-t Suprime la impresión de las cinco líneas de cabecera o de las cinco líneas de cola que se ponen normalmente en cada página.
-sc Sustituye los espacios de separación por el carácter «c».
-m Imprime todos los ficheros simultáneamente, uno por columna.

Es importante tener claro que la orden **pr** *no manda el resultado a la impresora,* sino que lo hace al fichero estándar de salida (normalmente la pantalla del terminal). Para que los ficheros ya formateados se impriman, puede redireccionar mediante un pipe la salida del comando **pr,** y mandárselo como entrada al comando **lpr,** es decir:

pr fichero | lpr

Algunos ejemplos de utilización del mandato **pr** son los siguientes:

```
% pr nombres

Jan  7 16:39 1987  nombres Page 1

mari
javier
juanjo
manolo
          .
          .   ( hasta la linea 66 en blanco)
          .
          .

% pr -t nombres
mari
javier
juanjo
manolo
%
```

En el listado anterior se ve claro cómo al utilizar **pr** sin ninguna opción hay cinco líneas de cabecera: dos en blanco, una línea con el texto de la cabecera y otras dos en blanco. Después de la línea 66 se añadirían otras cinco líneas de cola (todas en blanco). Al utilizar la opción **-t** desaparecen las líneas de cabecera (y también se eliminarían las de pie de página).

En caso de que se quiera obtener una salida en columnas se pone un número después del guión (opción **-n**, donde «n» es un número entero).

```
% pr -2 nombres

Jan  7 16:39 1987  nombres Page 1

mari                                    juanjo
javier                                  manolo

               .
               .
               .
               .

%
```

También pueden encadenarse opciones:

```
% pr -h "nombres de mis amigos" -2 nombres

Jan  7 16:39 1987  nombres de mis amigos Page 1

mari                                            juanjo
javier                                          manolo

                        .
                        .
                        .

%
```

En el siguiente ejemplo se sustituye el carácter de separación de palabras (un espacio en blanco) por el carácter «/».

```
% pr -2 -h "nombres de mis amigos" -s/ nombres

Jan  7 16:39 1987  nombres de mis amigos Page 1

mari/juanjo
javier/manolo

              .
              .
              .
              .

%
```

En caso de que se quiera sacar un número determinado de líneas por página (por ejemplo 3), y se desee conservar la cabecera y pie de página, se tendrá que poner 13 en la opción -l, ya que hay que añadir cinco líneas de cabecera y cinco de cola.

```
% pr -l13 nombres

Jan  7 16:39 1987  nombres Page 1

mari
javier
juanjo

Jan  7 16:39 1987  nombres Page 2

manolo
```

COPIA DE FICHEROS

Mediante la orden **cp** el sistema operativo UNIX permite copiar un fichero en otro. La sintaxis de esta orden es:

cp FicheroFuente FicheroDestino

El fichero «FicheroFuente» se copia sobre «FicheroDestino». Si «FicheroDestino» no existía antes de la copia, se crea con el modo de protección de «FicheroFuente». En caso de que ya existiese, se borra su anterior contenido para pasar a almacenar el contenido de «FicheroFuente», pero conserva el propietario y el modo de protección que tenía.

```
% cat quien
e0254    tty03   Jan  7 14:27
e0254    tty02   Jan  7 14:34
sec1     console Jan  7 15:25
luisjc   tty07   Jan  7 11:38
f0012    tty01   Jan  7 15:39
% ls -l quien
-rw-r--r--   1 e0254   2374        150 Jan  7 15:54 quien
% cp quien nuevoquien
% cat nuevo quien
e0254    tty03   Jan  7 14:27
e0254    tty02   Jan  7 14:34
sec1     console Jan  7 15:25
luisjc   tty07   Jan  7 11:38
f0012    tty01   Jan  7 15:39
% ls -l nuevoquien
-rw-r--r--   1 e0254   2374        150 Jan  7 16:57 nuevoquien
%
```

Con la orden **cp** también pueden copiarse uno o más ficheros en un directorio específico. La sintaxis es:

cp fichero(s) NombreDeDirectorio

```
% cp nomina.cob cobol
% ls -l cobol
total 9
-rwxrwxrwx   1 e0254    2374         4564 Jan  7 17:01 nomina.cob
%
```

BUSQUEDA DE UNA CADENA DE CARACTERES EN UN FICHERO

A veces es necesario buscar, en uno o varios ficheros, determinadas cadenas de caracteres. Para un fichero único la búsqueda puede realizarse con la ayuda de un programa editor, pero mientras éste busca en cada fichero separadamente, el sistema operativo UNIX nos proporciona una herramienta para búsquedas en múltiples ficheros simultáneamente de una manera más efectiva.

Esta herramienta es la orden **grep**, cuya sintaxis es:

grep [opciones] CadenaDeCaracteres Fichero(s)

Las opciones de este comando son las siguientes:

-v Muestra en la pantalla todas las líneas que *no* contienen la cadena de caracteres especificada.

-c Imprime sólo el número de líneas que contienen la cadena descrita.

-l Proporciona el nombre de los ficheros en que se encuentre al menos una vez la cadena de caracteres indicada.

-n Cada línea que contiene la cadena de caracteres especificada aparece precedida por su número de línea dentro de cada fichero.

Para evitar que el shell interprete los caracteres especiales de manera normal, deben ir precedidos del carácter «\» o se debe entrecomillar toda la expresión.

El siguiente ejemplo busca en el fichero «pruebaFORK.c» aquellas líneas que contengan la cadena de caracteres «printf».

```
% grep printf pruebaFORK.c
if (i==-1)  { printf("no puedo dividirme\n");  }   /* error */
                    printf("soy el padre y mi pid es %d \n",padrepid);
                    printf("el pid del hijo es %d \n",i);
                  printf("soy el hijo y mi pid es %d \n",hijopid);
%
```

Muestra en pantalla todas las líneas del mismo fichero que no contienen la cadena «printf».

```
% grep -v printf pruebaFORK.c
main()
/* prueba el fork,el wait, el exit  y el getpid */
{
int i;
int j;
int padrepid;
int hijopid;
i=fork();
   else if (i!=0)   {j= wait();
                      padrepid=getpid();
                    }
             else  {
                    hijopid=getpid();
                    exit(0);
                  }
}
%
```

Visualiza sólo el número de líneas que contienen la cadena «printf» de los ficheros «pruebaFORK.c» y «pruebaEXEC.c».

```
% grep -c printf pruebaFORK.c pruebaEXEC.c
pruebaFORK.c:4
pruebaEXEC.c:2
%
```

Visualiza numeradas las líneas del fichero «pruebaFORK.c» donde se encuentra la cadena «if».

```
% grep -n if pruebaFORK.c
9:if (i==-1)  { printf("no puedo dividirme\n");  }    /* error */
10:    else if (i!=0)    {j= wait();
%
```

Muestra en pantalla los nombres de los ficheros que contienen en alguna(s) línea(s) la cadena de caracteres «fork».

```
% grep -l fork pruebaFORK.c pruebaEXEC.c pruebaKILL.c
pruebaFORK.c
pruebaKILL.c
%
```

CLASIFICACION DE FICHEROS

Con la orden **sort** se pueden ordenar o clasificar uno o más ficheros línea a línea. También se pueden combinar para formar uno único. La sintaxis de este comando es:

sort [opciones] [+pos] [-pos] [-o salida] [-T FicheroFuente]

Un ejemplo sencillo de este comando puede ser el siguiente:

```
% cat quien
e0254    tty03    Jan  7 14:27
e0254    tty02    Jan  7 14:34
sec1     console  Jan  7 15:25
luisjc   tty07    Jan  7 11:38
f0012    tty01    Jan  7 15:39
% sort quien
e0254    tty02    Jan  7 14:34
e0254    tty03    Jan  7 14:27
f0012    tty01    Jan  7 15:39
luisjc   tty07    Jan  7 11:38
sec1     console  Jan  7 15:25
%
```

Algunas opciones de esta orden son:

-m	Fusiona los ficheros especificados.
-u	Elimina de la salida clasificada las líneas que estén duplicadas (consecutivas).
-d	Ordenación como diccionario: sólo cuentan letras, dígitos y espacios en blanco.
-f	Se ignora la distinción entre las letras mayúsculas y minúsculas.
-n	Clasifica primero el campo numérico.
-r	Clasifica en orden inverso.
-b	Ignora los espacios en blanco y los tabuladores en las comparaciones.
-c	Comprueba si el fichero ya está clasificado.
-o fichero	Guarda el resultado en «fichero».
-tx	El carácter «x» (puede ser cualquiera) es considerado como carácter de separación entre los distintos campos. Por defecto ese carácter es un espacio en blanco.

La orden **sort** trabaja teniendo en cuenta la línea, o campos dentro de ésta. Un campo es una cadena de caracteres no vacía, sin blancos, separada de otras cadenas de caracteres por blancos (a no ser que se especifique otra cosa con la opción **-tx**).

Las claves de clasificación se especifican mediante **+pos** y **-pos.** Eso quiere decir que la clave por la cual se va a clasificar u ordenar el(los) fichero(s) es un campo que empieza en **+pos** y termina justo antes de **-pos.** El primer campo de un fichero es +0, el segundo +1, el tercero +2, y así sucesivamente. Tanto **+pos** como **-pos** pueden expresarse con el formato **«m.n»,** donde **«m»** define el número de campos a saltar desde el inicio de la línea, y **«n»** el número de caracteres a saltar a partir de esta posición. La ausencia de **«n»** significa un 0, y la ausencia de **-pos** significa el final de la línea.

Por ejemplo, para ordenar el fichero «quién» tomando como clave el identificador del terminal, habría que teclear:

```
% sort +1b quien
sec1      console Jan   7 15:25
f0012     tty01   Jan   7 15:39
e0254     tty02   Jan   7 14:34
e0254     tty03   Jan   7 14:27
luisjc    tty07   Jan   7 11:38
%
```

+1b significa que se salta el primer campo (+0), que en este caso es el nombre de conexión de los usuarios, y también todos los espacios en blanco

que preceden al segundo campo, que es la clave. Normalmente la opción **-b** es de las más útiles.

En el ejemplo siguiente se clasifica el mismo fichero tomando como clave los dos últimos dígitos (los minutos de la hora de conexión), y la salida se almacena en el fichero «resultado».

```
% sort +4.3 -o resultado quien
% cat resultado
sec1      console Jan  7 15:25
e0254     tty03   Jan  7 14:27
e0254     tty02   Jan  7 14:34
luisjc    tty07   Jan  7 11:38
f0012     tty01   Jan  7 15:39
%
```

Para ver un ejemplo de fusión de ficheros suponga que ya existe el fichero «pares» y el fichero «impares», cuyos contenidos son los siguientes:

```
% cat pares
2
4
6
8
% cat impares
1
3
5
7
9
%
```

Para fusionarlos se emplea la opción **-m:**

```
% sort -m pares impares
1
2
3
4
5
6
7
8
9
%
```



COMPARACION DE FICHEROS

El comando **cmp** sirve para comparar dos ficheros cualesquiera. De manera opcional puede indicar las diferencias existentes y el lugar donde se han producido (línea y carácter dentro de la línea). La sintaxis de la orden es:

cmp [opción] fichero1 fichero2

Las opciones no se pueden concatenar, y son las siguientes:

-l Visualiza la localización en decimal del carácter diferente, y el valor octal de los caracteres para todos los caracteres diferentes.

-s No visualiza nada, pero devuelve el resultado de la comparación.

El ejemplo siguiente muestra la comparación entre los ficheros «pruebaFORK.c» y «pruebaEXEC.c»:

```
% cmp pruebaFORK.c pruebaEXEC.c
pruebaFORK.c pruebaEXEC.c differ: char 8, line 2
%
```

La orden **cmp** es útil cuando se copian ficheros, para comparar que ambas copias son idénticas. También se utiliza para comparar diferentes versiones de un mismo fichero, para saber si se han producido modificaciones de una versión a otra.

ORDENES PARA EL CONTROL DE PROCESOS DE USUARIO



EJECUCION DE UN PROCESO DE FORMA «DESATENTIDA»



NA característica importante del sistema operativo UNIX es que es un sistema operativo **multitarea,** es decir, permite la ejecución «simultánea» de varios **procesos** o trabajos. Esto significa que el ordenador estará ejecutando órdenes suyas mientras ejecuta también las de otro usuario. También es posible que un solo usuario ejecute varios trabajos a la vez. La manera de hacerlo es poniendo el signo «&» al final de la orden, la cual se ejecutará de manera «desatendida». Al hacer esto aparecerá en el terminal un número, e inmediatamente después el prompt del sistema. Ese número es el identificador del proceso, y a usted le sirve para preguntar la situación en la que se encuentra o para terminarlo antes de que él finalice por sí mismo.

Normalmente esta manera de trabajar es útil cuando se quiera realizar alguna tarea que lleva bastante tiempo en ejecutarse. Se consigue así que el usuario no tenga que esperar a que ese proceso termine, y pueda aprovechar el tiempo en la ejecución de otros trabajos.

```
% ls -l &
4365
%
```

ESTADO DE UN PROCESO DE USUARIO

Se pueden controlar uno o varios procesos que se están ejecutando de manera «desatendida» o **background** mediante el comando **ps.**

```
% ls -l | sort &
4386
4387
% ps
   PID TTY TIME COMMAND
  3567 04  0:39 csh
  4386 04  0:01 ls
  4387 04  0:00 sort
  4388 04  0:01 ps
%
```

Los títulos de cada columna significan lo siguiente:

PID	Identificador del proceso. Es el número que aparece en pantalla al ejecutar un trabajo en background.
TTY	Identificador del terminal.
TIME	Tiempo de ejecución que lleva el proceso.
COMMAND	Nombre del proceso.

Se puede ejecutar varias veces la orden para ver cómo evolucionan los procesos:

```
% ls -l | sort &
4414
4415
%ps
   PID TTY TIME COMMAND
  3567 04  0:40 csh
  4414 04  0:02 ls
  4415 04  0:00 sort
  4416 04  0:01 ps
% ps
   PID TTY TIME COMMAND
  3567 04  0:40 csh
  4414 04  0:04 ls
  4415 04  0:01 sort
  4417 04  0:01 ps
% ps
   PID TTY TIME COMMAND
  3567 04  0:41 csh
  4422 04  0:01 ps
%
```



ABORTAR UN PROCESO EN EJECUCION «DESATENDIDA»

Se puede abortar o finalizar a la fuerza un proceso normal pulsando «CRTL+C» o la tecla de interrupción (DEL), pero esto no sirve cuando el proceso se está ejecutando en background. Para acabar un proceso de este tipo se utiliza el comando **kill**, cuya sintaxis es:

kill NumeroDeProceso

donde «NumeroDeProceso» es el identificador del proceso (el valor que aparece en la columna PID al utilizar el comando **ps**).

```
% cp *.* directorio/. &
4292
% kill 4292
4292: cp: Terminated
% ps
   PID TTY TIME COMMAND
  3567 04  0:36 csh
  4293 04  0:01 ps
%
```



EJECUCION DE UN PROCESO EN UN INSTANTE ESPECIFICADO

Con el comando **at** puede ejecutarse una o un conjunto de órdenes en un día y hora específicos. La sintaxis de esta orden es la siguiente:

at hora [día] FicheroDeOrdenes

«Hora» es un número de uno a cuatro dígitos, con las opciones **A, P, N** o **M** para indicar AM, PM, mediodía o medianoche, respectivamente. Los números que tengan uno y dos dígitos son interpretados como una hora, y los números de tres y cuatro dígitos representan horas y minutos. Si a los números no les siguen letras se supone un reloj de veinticuatro horas (con lo que las opciones A, P, N o M son innecesarias).

Si no se especifica «día», se supone que la ejecución es en el día actual. En caso de que se indique, un día se puede representar por el nombre de un mes seguido del número del día (separados ambos por un espacio en

blanco), o puede ser el nombre de un día de la semana. Además, si se especifica la opción **week,** las órdenes a ejecutar lo harán una semana más tarde a partir de la fecha indicada.

Por último, «FicheroDeOrdenes» es el fichero (creado normalmente con un editor) donde las órdenes a ejecutarse están almacenadas.

Algunas posibles fechas podrían ser:

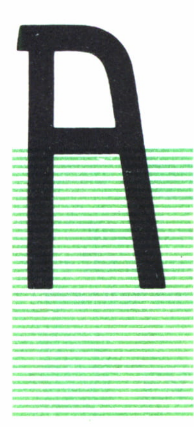
at 9am órdenes	Ejecuta «órdenes» hoy a las 9 de la mañana.
at 2230 órdenes	Ejecuta «órdenes» hoy a las 10:30 de la noche.
at 12N órdenes	Ejecuta «órdenes» hoy a las 12 del mediodía.
at 4PM apr 12 órdenes	Ejecuta «órdenes» el próximo 12 de abril a las 4 de la tarde.
at 715 mon week órdenes	Ejecuta «órdenes» una semana después del lunes que viene a las 7:15 de la mañana.

En el sistema UNIX los programas pendientes de **at** sólo se ejecutan en ciertos intervalos periódicos de tiempo. Esto puede ocurrir cada diez minutos, cada veinte, cada media hora o más, depende del sistema concreto.

UN EDITOR DE TEXTO: vi 7



INTRODUCCION



ALGO que nunca falta en un sistema operativo es el editor de texto, programa que tiene por función el ayudar al usuario a crear y modificar textos. En los sistemas operativos antiguos había editores llamados «de líneas», ya que la unidad de trabajo para ellos era una línea de texto, que se indica mediante un número. Estos editores son simples, pero requieren con frecuencia gran cantidad de comandos para realizar cualquier pequeña modificación, con lo que la complejidad del programa es bastante mayor que la potencia del mismo.

Otro segundo tipo de editores de textos más sofisticados son los llamados «de pantalla» o de página completa (full screen). El texto se ve como una serie continuada de caracteres, y se usan comandos de desplazamiento relativo (por caracteres, líneas o pantallas) para moverse por él. Otra característica importante es que visualizan instantáneamente el efecto de cualquier modificación.

Los dos tipos de editores utilizan una copia del fichero en memoria sobre la cual se trabaja. Cuando acaba la sesión el contenido de la memoria se copia hacia el disco para actualizar el fichero.

El sistema operativo UNIX pone a disposición de los usuarios dos editores principales: **ed**, un editor «de líneas», y **vi**, un editor «de pantalla», que es el que se estudia en este capítulo.

Uno de los aspectos que hacen más agradable el uso de **vi** frente a **ed** es que se tiene en todo momento una ventana de pantalla sobre el texto que se edita. Esta ventana puede desplazarse por el texto hacia arriba, hacia abajo, línea a línea o pantalla a pantalla, según los comandos utilizados. El cursor indica en todo momento la posición del texto donde se encuentra el usuario.

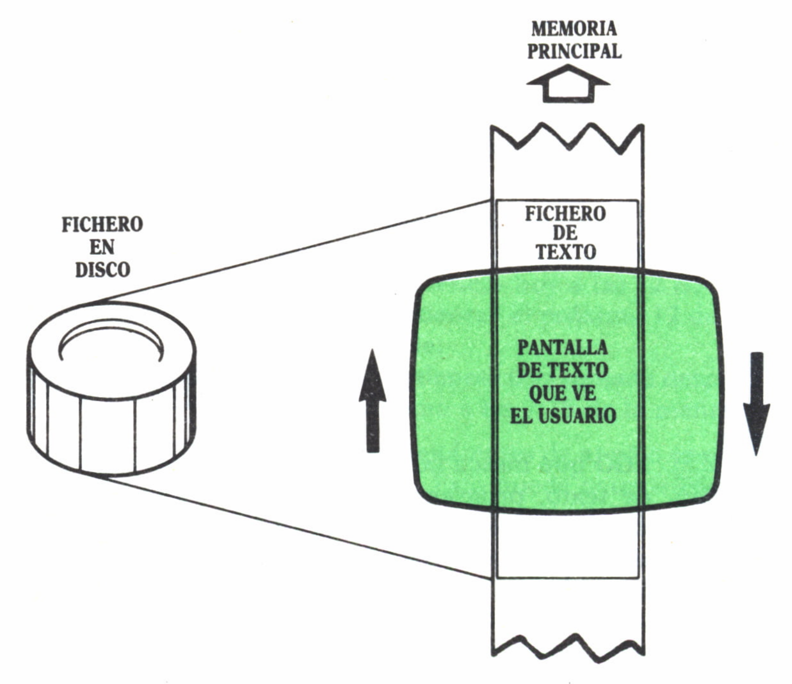


Fig. 7.1. Esquema conceptual de editor de texto vi.



MODOS DE FUNCIONAMIENTO DE vi

Para entender bien el uso de este editor es fundamental saber en todo momento en qué modo de funcionamiento se encuentra. El editor **vi** dispone de tres modos:

— modo comando
— modo inserción
— modo ex



Modo comando

Se entra en este modo cuando se empieza una sesión de edición. La mayoría de las teclas tienen asociado un comando específico, e incluso las minúsculas y las mayúsculas tienen casi siempre un significado diferente. Por ejemplo, **a** sirve para añadir texto, **r** sustituye un carácter, **R** sustituye texto, etc. Hay cerca de 90 comandos a disposición del usuario, los cuales se ejecutan sin que su nombre aparezca en el terminal (no producen eco de

pantalla). A veces los comandos llevan parámetros y pueden ir precedidos de un factor de repetición.



Modo inserción

Se entra en este modo después de teclear ciertos comandos, como **i, a, o,** y algunos más. En modo inserción los caracteres tecleados se introducen en el texto, razón por la cual este modo de funcionamiento del editor tiene pocos comandos (principalmente para corregir los errores tipográficos).

Con la pulsación de la tecla <**ESC**> se vuelve al modo comando.



Modo ex

Sus comandos aparecen sobre la última línea de la pantalla. Algunos de los mismos permiten volver al sistema (bajo el control de la shell) para ejecutar comandos UNIX.



ENTRADA EN EL EDITOR vi Y VUELTA AL SISTEMA

La manera de invocar al editor desde la shell es mediante el comando **vi,** que tiene la siguiente sintaxis:

vi [opción] [posicionamiento] fichero

Las opciones son:

-r Recupera la antigua versión del fichero salvaguardada después de una caída del sistema.

-R Impide la modificación del texto.

El **posicionamiento** se utiliza para situarse directamente sobre una línea del fichero, o sobre una cadena de caracteres determinada. Puede tener la forma:

+n Posiciona el cursor en la línea «n» del fichero.

+/cadena Posiciona el cursor en la primera ocurrencia de la cadena de caracteres indicada en «cadena».

Fichero indica el fichero de texto sobre el que se quiere trabajar.

Al invocar al editor **vi** el sistema responde borrando toda la pantalla y mostrando en la parte inferior de la misma una línea de control, que indica si el fichero indicado en la llamada es nuevo o proporciona su tamaño en caso de que ya esté creado.

Se puede volver al sistema de tres formas:

1. Con el comando **ZZ,** que guarda el fichero en disco en caso de que se haya modificado.
2. Con **:wq** (write, quit), que escribe el buffer de trabajo sobre el disco.
3. Con **:q!** (quit), que vuelve al sistema sin salvar nada en disco.



COMANDOS DE vi

Posicionamiento en pantalla

Para simplificar la notación de los comandos, se utiliza el símbolo «ˆ» para indicar la tecla de CTRL. Por ejemplo, **«ctrl + D»** se indicaría como **«ˆD».**

Para moverse dentro del texto se utilizan los siguientes comandos:

ˆD	Avanza media pantalla hacia adelante.
ˆU	Avanza media pantalla hacia detrás.
ˆF	Avanza una pantalla hacia adelante.
ˆB	Avanza una pantalla hacia detrás.
«flecha hacia arriba»	Sube el cursor una línea.
«flecha hacia abajo»	Baja el cursor una línea.
nG	Salta a la línea «n».
+	Posiciona el cursor al principio de la línea siguiente. Equivale a <CR> o <RETURN>.
-	Posiciona el cursor al principio de la línea anterior.
H	Sitúa el cursor en la primera línea.
M	Sitúa el cursor en medio de la pantalla.
L	Sitúa el cursor en la última línea de la pantalla.
n**H** n**M** n**L**	Avanza «n» líneas a partir de la referencia.
w	Avanza a la siguiente palabra.
b	Retrocede a la palabra anterior.
W	Avanza a la siguiente palabra saltándose las puntuaciones.
B	Retrocede a la palabra anterior saltándose las puntuaciones.
e	Salta al fin de la palabra actual.
fc	La «c» significa cualquier carácter. Con este comando se salta al siguiente «c» de la línea actual.
Fc	Salta al anterior «c» de la línea actual.
ˆ	Salta al principio de la línea actual (ese carácter no es la tecla CTRL).
$	Salta al final de la línea actual.
)	Salta al final de la sentencia actual.

B Salta al final del párrafo actual.
]] Salta al final de la sección actual.
(Salta al inicio de la sentencia actual.
A Salta al inicio del párrafo actual.
[[Salta al inicio de la sección actual.



Inserción de texto

i Inserta a partir de la posición actual.
I Inserta al principio de la línea actual.
a Añade después de la posición actual.
A Añade detrás de la línea actual.
o Crea una nueva línea después de la actual.
O Crea una nueva línea antes de la actual.



Borrado de texto

x Borra el carácter sobre el cual está situado el cursor.
X Borra el carácter que precede al cursor.
n**x** n**X** Borra «n» caracteres de las dos formas indicadas.
dw Borra una palabra.
db Borra una palabra hacia atrás.
dd Borra una línea (la que tiene el cursor).
n**dw** Borra «n» palabras.
n**db** Borra «n» palabras hacia atrás.
n**dd** Borra «n» líneas.
dH Borra hasta la primera línea.
dM Borra hasta la mitad de la pantalla.
dL Borra hasta la última línea de la pantalla.
dfc Borra hasta la primera «c» (cualquier carácter) de la línea.
dFc Borra hacia atrás hasta la primera «c» de la línea.
dtc Borra hasta la primera «c» exclusive.
dTc Borra hacia atrás hasta la primera «c» exclusive.
d) Borra el resto de la sentencia actual.
dB Borra el resto del párrafo actual.
d]] Borra el resto de la sección actual.
d(Borra hacia atrás el resto de la sentencia actual.
dA Borra hacia atrás el resto del párrafo actual.
d[[Borra hacia atrás el resto de la sección actual.
J Borra el cambio de línea de la actual.
p Reescribe a continuación lo borrado.
P Reescribe hacia atrás lo borrado.



Búsqueda

/cadena Busca la primera aparición de la cadena de caracteres especificada en «cadena».

?cadena Busca la anterior aparición de la cadena de caracteres especificada en «cadena».

Si después de cualquiera de estos comandos se pulsa:

n Busca la siguiente aparición de la cadena de caracteres indicada anteriormente.

^ Busca en principios de línea la cadena de caracteres indicada anteriormente.

$ Busca en finales de línea.



Sustitución y modificación de textos

rx	Reemplaza el carácter sobre el cual está el cursor por el carácter representado por «x».
R	Sustituye varios caracteres a la vez.
cw	Cambia una palabra.
cc	Cambia una línea.
c$	Cambia hasta el final de la línea.
g/cadena1/s/cadena2/g	Sustituye la cadena de caracteres «cadena1» por «cadena2». La opción **g** del final indica que se quieren modificar todas las ocurrencias en el texto de esa cadena.
g/texto/d	Borra las líneas donde aparezca el texto indicado.



Otros comandos

. Repite la última operación.

u Deshace el último cambio realizado.

U Deshace los cambios realizados en la línea actual.



COMANDOS DEL MODO ex

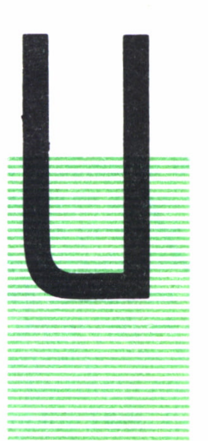
Para pasar al modo **ex** se escribe **«:»** seguido por el comando y sus argumentos. Este tipo de comandos permiten trabajar sobre otros ficheros, leer y escribir ficheros.

Entre todos los comandos cabe destacar los siguientes:

:w fichero	Escribe el contenido del buffer de edición en el fichero cuyo nombre se pasa como parámetro.
:w! fichero	Fuerza la escritura del contenido del buffer de edición en «fichero», aunque éste ya existiera.
:w>> fichero	Agrega el contenido del buffer de edición sobre el fichero nombrado.
:wq	Guarda el fichero en disco y sale del editor (se devuelve el control a la shell).
:q	Deja la edición sin salvar (**:q!** para forzar).
:! comando-UNIX	El signo «!» permite salir temporalmente de **vi**. Después de ejecutarse el comando UNIX especificado, se vuelve al editor.
:sh	Cede control a la shell. Pulsando **^d** se retorna a **vi**.

UNA VISION INTERNA DE UNIX





ESTRUCTURA GENERAL DE UNIX



UNIX no es sólo un sistema operativo, sino que es un entorno entero de programación. Lo constituyen varias partes:

— Un sistema operativo propiamente dicho, al que se le suele llamar núcleo UNIX. Implementa los procesos de usuario y un sistema de ficheros residente en disco. Ejecuta las llamadas al sistema de los procesos de usuario. Maneja las interrupciones y controla los periféricos.

— Un intérprete de comandos (la shell).

— Las utilidades, entre las que se pueden encontrar:

- un compilador de lenguaje C **(cc)**
- editores de texto **(vi, ed)**
- un procesador de textos **(nroff)**
- un ensamblador
- herramientas para la escritura de compiladores
- compiladores de diferentes lenguajes de programación (Pascal, COBOL, FORTRAN 77...).

El usuario tiene tres vías o medios de acceso para comunicarse con el sistema:

— A través de la shell o intérprete de mandatos.

— Mediante llamadas en un programa de usuario a primitivas del sistema. Cada primitiva es una función o un procedimiento del núcleo que realiza una misión específica.

— Mediante el conjunto de procedimientos de la biblioteca de ejecución («run time library»).

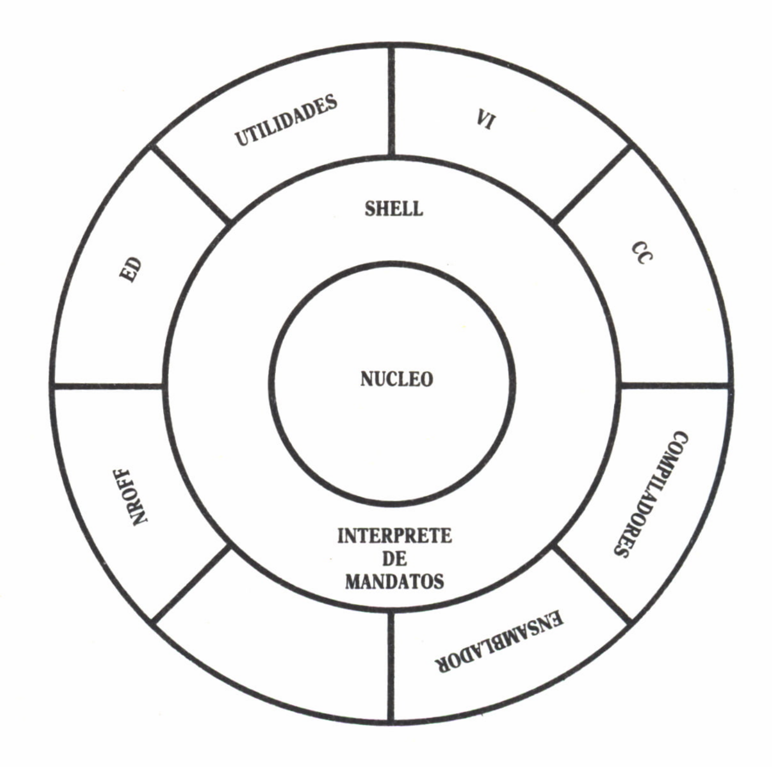


Fig. 8.1 Esquema de la arquitectura del sistema UNIX.



PROCESOS EN UNIX

Concepto de proceso

Un programa es una colección de instrucciones que describen un **proceso.** Un proceso es una actividad asíncrona. Puede interpretarse como la ejecución de las instrucciones del programa por una CPU en un área de memoria determinada.

El concepto de proceso engloba no sólo la ejecución de unas instrucciones, sino todo el entorno que constituye el ambiente de ejecución; es decir, una pila, un segmento de datos, el estado de los registros del procesador, el estado de los ficheros abiertos, el directorio actual...

El sistema es dinámico; en general, un proceso no tiene conocimiento completo de la existencia de otros.

Cada proceso de usuario tiene en UNIX su propia memoria virtual o espacio de direccionamiento, al cual sólo puede acceder el propio proceso.

El espacio de direccionamiento de un proceso consiste en tres partes o segmentos:

1. Segmento de código fuente.
2. Segmento de datos.
3. Segmento de pila.

Un proceso de usuario está identificado por un número de proceso único. Otros procesos no necesitan saber la posición real del que está en memoria, pero pueden referirse a él por su nombre (número de proceso único). Se mantiene un descriptor para cada proceso de usuario con su estado de proceso y su localización en memoria.

Estos procesos de usuario comparten la CPU cíclicamente entre todos los que están activos. Cada uno tiene asignada una rodaja de tiempo. Cuando se acaba la misma, el proceso es interrumpido y los valores de sus registros se almacenan en una estructura de datos de usuario.

Los procesos de usuario se crean a petición de otros procesos de usuario a través del código de la primitiva **fork.** Después de la ejecución de **fork** existen en el sistema dos procesos, de los cuales uno (el que ejecuta la primitiva) es el padre del proceso que se crea. Ambos ejecutan el mismo código con los mismos datos, pero de manera totalmente independiente. Lo único que difiere a estos dos procesos es el valor que devuelve la función **fork.** En el caso del hijo el valor de retorno es cero, y en el caso del padre es un número positivo que identifica al hijo creado. Ese número es el **pid** (identificador de proceso). En caso de que se produzca un error y no sea posible crear al hijo, la primitiva fork devuelve al padre un valor negativo.

```
main()
/* programa que prueba la primitiva FORK */
{
int i;
int PadrePid;      /* identificativo del proceso padre */
int HijoPid;       /* identificador del proceso hijo */
i=fork();
if (i==-1)    {    printf(" ERROR EN LA DIVISION \n");    }
   else if (i!=0) {
                    /*  esto lo ejecuta el proceso padre   */
                    PadrePid=getpid(); /* obtiene su ident. de proceso */
                    printf(" SOY EL PADRE Y MI PID ES %d \n",PadrePid);
                    printf(" EL IDENTICADOR DE MI HIJO ES %d \n",i);
                   }
             else  {
                    /* esto sólo lo ejecuta el proceso hijo */
                    HijoPid=getpid();  /* obtiene su ident. de proceso */
                    printf(" SOY EL HIJO Y MI PID ES %d \n",HijoPid);
                   }
```



Comunicación entre procesos

Los procesos no comparten la memoria y, por tanto, no la pueden utilizar como medio de comunicación. Con frecuencia se comparte el segmento de código, pero está protegido contra escritura, y el único objetivo de esta compartición es el de obtener una mejor utilización de la memoria, de manera que una sola copia del código esté eventualmente ejecutada por varios procesos.

El sistema operativo UNIX ofrece como medio de comunicación entre procesos un tipo de fichero especial llamado **pipe.** El **pipe** permite la transferencia de datos entre procesos, utilizando para ello una política FIFO. Esto quiere decir que el orden en que los datos son escritos en el **pipe** es el mismo que el orden en que son leídos de él.

El número de procesos leyendo de un **pipe** no tiene necesariamente que ser igual al número de procesos escribiendo en él. Además, la comunicación se efectúa de manera que los procesos que están en un extremo del **pipe** no saben cuántos procesos hay en el otro extremo.

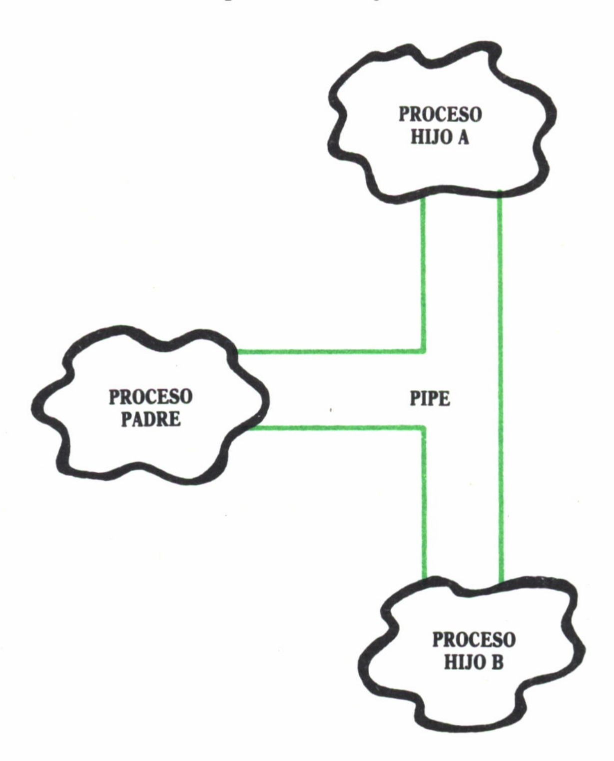


Fig. 8.2. Conexiones a un pipe de un proceso padre y dos procesos hijos.

Un **pipe** se crea mediante la llamada al sistema **«pipe».** Cuando se ejecuta se crean dos descriptores de ficheros: uno de lectura, que indica por dónde va la lectura del **pipe,** y otro de escritura para indicar la posición donde se efectuará la siguiente escritura. Conviene recordar en este punto que un **pipe** es un fichero, y que para UNIX un fichero es sólo una secuencia de caracteres.

Para ver un ejemplo de comunicación entre procesos supongamos que un proceso padre (la shell, por ejemplo) comunica dos procesos hijos entre sí, uno de los cuales es un productor y el otro un consumidor. Primero el padre debe crear el **pipe,** y luego, mediante dos llamadas a la primitiva **fork,** crear los dos hijos, los cuales heredan las conexiones (los descriptores de lectura y escritura) del padre, pero sigue habiendo un solo ***pipe.*** Gráficamente quedaría algo similar a la figura 8.2.

Como un hijo es productor y el otro es consumidor, se eliminan los descriptores que ya no sirven de nada, es decir, algo parecido a cerrar las conexiones inútiles, con lo que los dos procesos hijos quedarían conectados como muestra la figura 8.3.

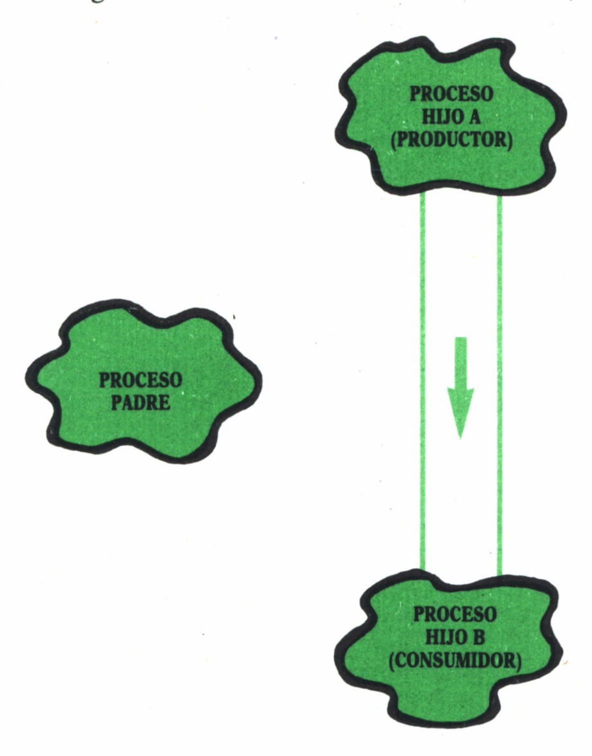


Fig. 8.3 Conexiones a un pipe de un proceso hijo productor y otro proceso hijo consumidor.

Sincronización de procesos

La cooperación entre distintos procesos implica la existencia de algún mecanismo de sincronización para que esa cooperación se efectúe de forma correcta.

Una forma importante de sincronización es la que se realiza entre un proceso padre y un proceso hijo mediante la llamada a la primitiva **wait.** El proceso que ejecuta esta función del sistema espera a que finalice uno de sus hijos. Como retorno, la función proporciona la identificación del proceso hijo finalizado y el estado de terminación del mismo.

Es importante dejar claro que durante un **wait** el proceso padre espera a que finalice alguno de sus hijos, no importa cuál. El proceso padre tiene conocimiento de que algún hijo ha acabado sólo si está esperando en un **wait,** y necesitará tantas llamadas a esta primitiva como procesos hijos deba esperar.

El ejemplo siguiente muestra en lenguaje C cómo un proceso padre espera a que los dos hijos que ha creado ejecuten algo.

```
main()
/* programa que prueba la primitiva WAIT */
{
int i;
int j;
int PadrePid;   /* identificativo del proceso padre */
int HijoPid;    /* identificador del proceso hijo */
i=fork();
if (i==-1)   {   printf(" ERROR EN LA DIVISION \n");   }
    else if (i!=0) {
                    /*  esto lo ejecuta el proceso padre  */
                    j=wait();   /* proceso padre espera alhijo */
                    PadrePid=getpid(); /* obtiene su ident. de proceso */
                    printf(" SOY EL PADRE Y MI PID ES %d \n",PadrePid);
                    printf(" EL IDENTICADOR DE MI HIJO ES %d \n",i);
                   }
            else   {
                    /* esto sólo lo ejecuta el proceso hijo */
                    HijoPid=getpid();  /* obtiene su ident. de proceso */
                    printf(" SOY EL HIJO Y MI PID ES %d \n",HijoPid);
                   }
```



SISTEMA DE FICHEROS

Distribución de los datos en disco

Los ficheros UNIX residen en disco. Cada disco se divide en cinco áreas:

DISCO

BLOQUE DE ARRANQUE: UNIX se inicializa cargando y ejecutando este bloque.
SUPERBLOQUE: Especifica límites de las tres áreas siguientes en el disco. También contiene la cabeza de la lista de bloques libres disponibles para ser asignados a ficheros.
AREA DE I-NODOS: Contiene descriptores (i-nodos) para cada fichero o directorio del disco. Cada i-nodo es del mismo tamaño (64 bytes en UNIX versión 7). El segundo i-nodo representa el directorio raíz del disco.
AREA DE FICHEROS: Se usa para almacenar el contenido de los ficheros.
AREA DE INTERCAMBIO: Contiene procesos de usuario cuando han sido sacados de memoria central.

Algunas instalaciones UNIX hacen que un disco físico contenga más de un disco lógico, con lo que cada disco lógico tendrá las áreas ya mencionadas.



Sistema de ficheros plano y en árbol

La implementación del sistema de ficheros UNIX divide el sistema de ficheros en dos capas:

— La más baja, el **sistema plano de ficheros** (o i-node file system), llamada así porque no tiene directorios. Cada fichero tiene un número: 1, 2, 3... Cada número dice qué i-nodo define al fichero.

— La capa más alta: el **sistema de ficheros en árbol.** Esta capa usa el sistema plano de ficheros para implementar el sistema de ficheros en árbol de UNIX. Aquí se usa un fichero plano para representar:

- un directorio del sistema jerarquizado UNIX.
- un fichero ordinario del usuario.



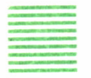
Formato de los directorios

Un directorio es un fichero plano y consiste en entradas (líneas) de 16 bytes. Cada entrada consiste en:

— 2 bytes: número de i-nodo.
— 14 bytes: nombre del fichero.

En cada directorio se tienen siempre dos entradas, que son:

— dos puntos (..), que corresponde al padre de ese directorio.
— un punto (.), que representa el directorio actual, es decir, es una referencia a sí mismo.



Formato de los i-nodos

El contenido de un fichero no se queda junto con su información de control, es decir, el fichero y su i-nodo no están juntos en el disco.

Cada i-nodo consiste en los siguientes campos de información:

NUMERO DE USUARIO: Nombre de la cuenta del usuario.
NUMERO DE GRUPO: Utilizado para protecciones de los ficheros.
PROTECCIONES: Cadena de los bits de protección.
TIEMPOS: Creación del i-nodo, última lectura y última actualización del fichero.
CODIGO DEL FICHERO: Especifica si el i-nodo representa un directorio, un fichero ordinario de usuario o un fichero especial.
TAMAÑO: Longitud del fichero en bytes.
LISTA DE BLOQUES: Localiza el contenido del fichero.
NUMERO DE ENLACES: Número de directorios referenciando ese i-nodo.

Todos estos campos, excepto el código del fichero y el número de enlaces, pueden ser considerados como del sistema plano de ficheros. Es responsabilidad del sistema de ficheros en árbol decidir si un i-nodo representa un directorio, y registar cuantas entradas de directorio referencian (están enlazadas a) un directorio particular.

Si el código del fichero representa un fichero especial el i-nodo no representa un fichero en disco, sino que está siendo usado para acceder a un dispositivo periférico.



Lista de bloques

La lista de bloques de un i-nodo consiste en punteros a bloques en el área de ficheros de disco.

Los primeros 10 punteros localizan directamente los bloques de disco que contienen el fichero. Cada bloque de disco tiene 512 bytes (el UNIX de la Universidad de Berkeley está implementado con bloques de 1024 bytes), luego estos 10 punteros localizan los primeros 5120 bytes del fichero, para los cuales sólo se necesita un acceso a disco.

El puntero 11 es usado cuando el fichero excede de 5120 bytes, y localiza un bloque de disco, el cual contiene punteros a bloques que sí contienen ya datos del fichero. Así, los bytes del fichero que son localizados mediante el puntero 11 requieren una referencia más a disco. Utilizando el puntero 11 podemos acceder al byte 70656 (69 K) del fichero.

Más allá del byte 70656 necesitamos usar el puntero 12, que ya es una doble indirección, ya que localiza un bloque conteniendo punteros a bloques, los cuales, a su vez, contienen punteros a bloques de datos del fichero. Con el puntero 12 referenciamos hasta 8261 K del fichero.

Más allá de ese tamaño usamos el puntero 13, el cual ya es una triple indirección, y soporta un tamaño máximo de fichero de ¡1056 Mbytes!

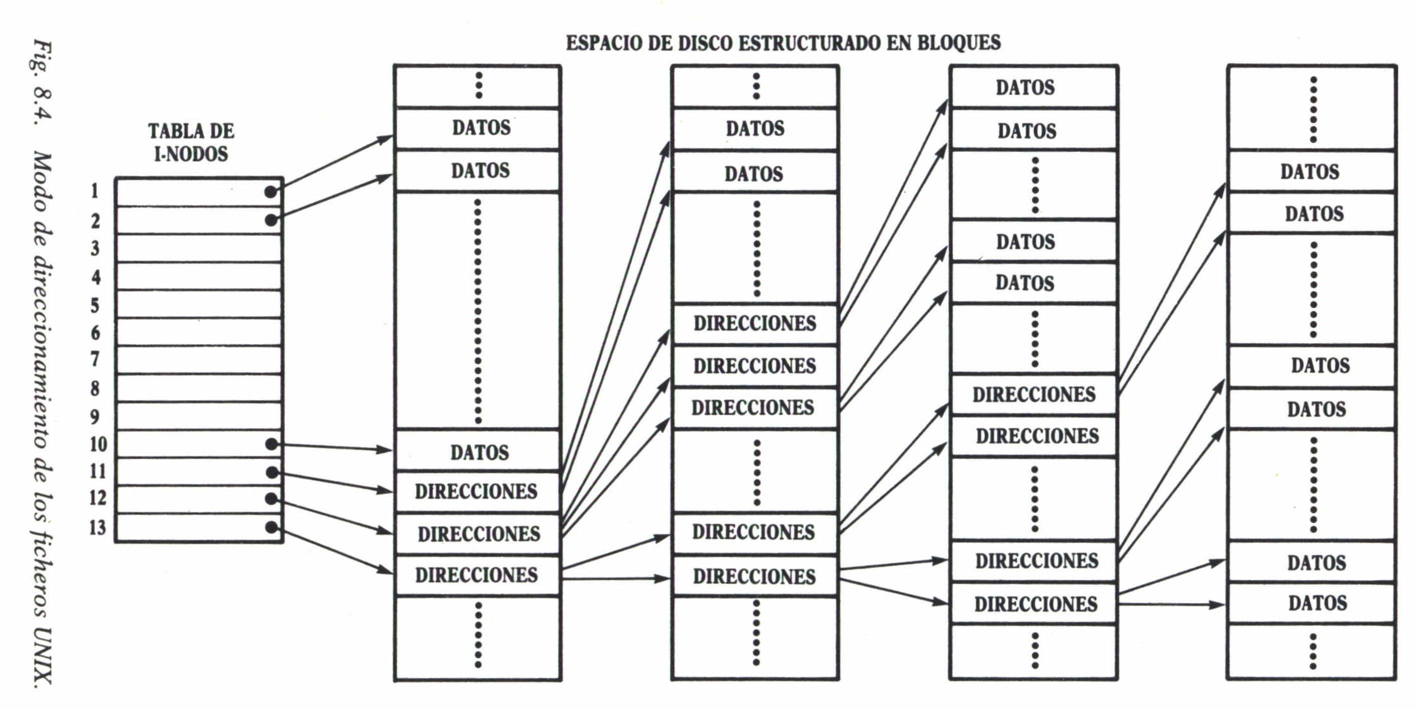


Fig. 8.4. Modo de direccionamiento de los ficheros UNIX.

APENDICE
LLAMADAS AL SISTEMA
EJECUTABLES EN C

ccess	Determina la accesibilidad de un fichero.
acct	Produce los ficheros de contabilidad del sistema.
alarm	Produce una señal después de un tiempo especificado.
chdir	Cambia el directorio por defecto.
chmod	Cambia las protecciones de un fichero.
close	Cierra un fichero.
creat	Crea un nuevo fichero.
dup	Crea un segundo descriptor para un fichero.
exec	Ejecuta un fichero a partir de otro que está ejecutándose.
exit	Finaliza un proceso en ejecución.
fork	Crea una copia (un proceso hijo) del mismo proceso.
getuid	Proporciona la identidad de un usuario.
getpid	Proporciona el identificador de un proceso.
kill	Aborta la ejecución de un proceso.
link	Produce un enlace de un fichero.
lseek	Permite acceder directamente a una posición de un fichero.
mknod	Añade un fichero a un directorio.
nice	Asigna una prioridad de ejecución a un proceso.
open	Abre un fichero.
pause	Detiene un proceso en espera de una señal.
pipe	Crea un pipe entre dos procesos.
setuid	Cambia la identificación de un usuario.
setgid	Cambia la identificación de un grupo.
signal	Espera o ignora una señal.
time	Proporciona la fecha y la hora.
times	Proporciona el tiempo de ejecución de un proceso.
unlink	Borra una entrada de un directorio.
unlimit	Indica el número máximo de ficheros que pueden crearse.
umask	Crea o modifica la máscara de protecciones.
wait	Espera el fin de la ejecución de un proceso.

BIBLIOGRAFIA

R. THOMAS: *Sistema Operativo UNIX. Guía del usuario.* McGraw Hill, 1984.
A. B. FONTAINE y PH. HAMMES: *UNIX, Sistema y entorno.* Masson, 1984.
H. LUCAS, B. MARTIN y DE SABLET: *Sistema Operativo UNIX.* Paraninfo, 1986.
M. J. BACH: *The design of the UNIX Operating System.* Prentice Hall Inc.
R. C. HOLT: *Concurrent Euclid, The UNIX System and Tunis.* Addison-Wesley, 1983.
J. L. PETERSON y A. SILBERSCHATZ: *Operating System Concepts.* Addison-Wesley, 1983.
P. B. HANSEN: *Operating System Principles.* Prentice Hall, Inc. 1973.
Informática, tomo 2. Ediciones Ingelek.
Tu micro personal, número 1, febrero 1986. Ediciones Ingelek.
Micros, número 7, mayo 1984. Ediciones Arcadia.

ENCICLOPEDIA PRACTICA DE LA

INFORMATICA APLICADA

INDICE GENERAL

5 AÑADA PERIFERICOS A SU ORDENADOR

Breve descripción de varios periféricos que facilitan la comunicación con el ordenador personal, con algunos ejemplos de fácil construcción: ratón, lápiz óptico, marco para pantalla táctil...

6 GRAFICOS ANIMADOS CON EL ORDENADOR

En este libro las técnicas utilizadas para la animación son el resultado de unas pocas ideas básicas muy sencillas de comporender. Descubrirá los trucos y secretos de movimientos, choques, rebotes, explosiones, disparos, saltos, etc.

7 JUEGOS INTELIGENTES EN MICROORDENADORES

Los ordenadores pueden enfrentarse de forma «inteligente» ante puzzles y otros tipos de juegos. Esto es posible gracias al nuevo enfoque que ha dado la IA a la tradicional teoría de juegos.

8 PERIFERICOS INTERACTIVOS PARA SU ORDENADOR

Descripción detallada de la forma de construir, paso a paso y en su propia casa, dispositivos electrónicos que aumentarán la potencia y facilidad de uso de su ordenador: tableta digitalizadora, convertidores de señales analógicas, comunicaciones entre ordenadores.

9 COMO HACER DIBUJOS TRIDIMENSIONALES EN EL ORDENADOR PERSONAL

Compruebe que también con su ordenador personal puede llegar a diseñar y calcular imágenes en tres dimensiones con técnicas semejantes a las utilizadas por los profesionales del dibujo con equipos mucho más sofisticados.

10 PRACTIQUE MATEMATICAS Y ESTADISTICA CON EL ORDENADOR

En este libro se repasan los principales conceptos de las Matemáticas y la Estadística, desde un punto de vista eminentemente práctico y para su aplicación al ordenador personal. Se basan los diferentes textos en la presentación de pequeños programas (que usted podrá introducir en su ordenador personal).

11 CRIPTOGRAFIA: LA OCULTACION DE MENSAJES Y EL ORDENADOR

En este libro se presentan las técnicas de mensajes a través de la criptografía desde los primeros tiempos hasta la actualidad, en que el uso de los computadores ha proporcionado la herramienta necesaria para llegar al desarrollo de esta técnica.

12 APL: LENGUAJE PARA PROGRAMADORES DIFERENTES

APL es un lenguaje muy potente que proporciona gran simplicidad en el desarrollo de programas y al mismo tiempo permite programar sin necesidad de conocer todos los elementos del lenguaje. Por ello es ideal para quienes reúnan imaginación y escasa formación en Informática.

13 ECONOMIA DOMESTICA CON EL ORDENADOR PERSONAL

Breve introducción a la contabilidad de doble partida y su aplicación al hogar, con explicaciones de cómo utilizar el ordenador personal para facilitar los cálculos, mediante un programa especialmente diseñado para ello.

14 COMO SIMULAR CIRCUITOS ELECTRONICOS EN EL ORDENADOR

Introducción a los diferentes métodos que se pueden emplear para simular y analizar circuitos electrónicos, mediante la utilización de diferentes lenguajes.

15 COMO CONSTRUIR SU PROPIO ORDENADOR

Cuando se trabaja con un ordenador, lo único que puede apreciarse, a simple vista, es una especie de caja negra que, misteriosamente, acepta una serie de instrucciones. En realidad, un ordenador es una máquina capaz de recibir, transformar, almacenar y suministrar datos.

16 EL ORDENADOR COMO INSTRUMENTO MUSICAL Y DE COMPOSICION

Análisis de cómo se puede utilizar el ordenador para la composición o interpretación de música. Libro eminentemente práctico, con numerosos ejemplos (que usted podrá practicar en su ordenador casero) y lleno de sugerencias para disfrutar haciendo de su ordenador un verdadero instrumento musical.

17 SISTEMAS OPERATIVOS: EL SISTEMA NERVIOSO DEL ORDENADOR

Características de diversos sistemas operativos utilizados en los ordenadores personales y caseros. Se trata de llegar al conocimiento, ameno aunque riguroso, de la misión del sistema operativo de su ordenador, para que usted consiga sacar mayor rendimiento a su equipo.

18 UNIX, EL ESTANDAR DE LOS SISTEMAS OPERATIVOS MULTIUSUARIO

La aparición y posterior difusión del sistema operativo UNIX supuso una revolución en el mercado, de tal modo que se ha convertido en el estándar de los sistemas multiusuario. Su aparente complejidad podría provocar, en principio, un primer rechazo, pero debido a su potencia se convierte rápidamente en una extraordinaria herramienta de trabajo apta para cualquier tipo de aplicaciones.

19 EL ORDENADOR Y LA ASTRONOMIA

Los cálculos astronómicos y el conocimiento del firmamento en un libro apasionante y curioso.

20 VISION ARTIFICIAL. TRATAMIENTO DE IMAGENES POR ORDENADOR

El procesado de imágenes es un campo de reciente y rápido desarrollo con importantes aplicaciones en área tan diversas como la mejora de imágenes biomédicas, robóticas, teledetección y otras aplicaciones industriales y militares. Se presentan los principios básicos, los sistemas y las técnicas de procesado más usuales.

21 PRACTIQUE HISTORIA Y GEOGRAFIA CON SU ORDENADOR

Libro interesante para los aficionados a estas ciencias, a quienes presenta una nueva visión de cómo utilizar el microordenador en su estudio.

22 LA CREATIVIDAD EN EL ORDENADOR. EXPERIENCIAS EN LOGO

El LOGO es un lenguaje enormemente capacitado para la creación principalmente gráfica y en especial para los niños. En este sentido se han desarrollado numerosas experiencias. En el libro se analizan estas experiencias y las posibilidades del LOGO en este sentido, así como su aplicación a su ordenador casero para que usted mismo (o con sus hijos) pueda repetirlas.

23 EL LENGUAJE C, PROXIMO A LA MAQUINA

Lenguaje de programación que se está imponiendo en los microordenadores más grandes, tanto por su facilidad de aprendizaje y uso, como por su enorme potencia y su adecuación a la programación estructurada. Vinculado íntimamente al sistema operativo UNIX es uno de los lenguajes de más futuro entre los que se utilizan los micros personales.

24 COMO ELEGIR UNA HOJA ELECTRONICA DE CALCULO

En este título se estudian las diferentes versiones existentes de esta aplicación típica, desde el punto de vista de su utilidad para, en función de las necesidades de cada usuario y del ordenador de que dispone, poder elegir aquélla que más se adecúe a cada paso.

25 PRACTIQUE FISICA Y QUIMICA CON SU ORDENADOR

Libro eminentemente práctico para realizar pequeños «experimentos» con su ordenador y distraerse de un modo útil.

26 EL ORDENADOR Y LA LITERATURA

En este libro se examinan procesadores de textos, programas de análisis literario y una curiosa aplicación desarrollada por el autor: APOLO, un programa que compone estructuras poéticas.

27 PRACTIQUE CIENCIAS NATURALES CON EL ORDENADOR

Ejemplos sencillos para practicar con el ordenador. Casos curiosos de la Naturaleza en forma de programas para su ordenador personal.

28 ERGONOMIA: COMUNICACION EFICIENTE HOMBRE-MAQUINA

Análisis de la comunicación entre el hombre y la máquina, y estudio de diferentes soluciones que tienden a facilitarla lo más posible.

29 LOS LENGUAJES DE LA INTELIGENCIA ARTIFICIAL

Libro en que se describen los lenguajes específicos para la «elaboración del saber» y los entornos de programación correspondientes. El conocimiento de estos lenguajes, además de interesante en sí mismo, es sumamente útil para entender todo lo que la Informática Artificial supondrá para el futuro de la Informática.

30 ¿MAQUINAS MAS EXPERTAS QUE LOS HOMBRES?

Después de situar los «sistemas expertos» en el contexto de la inteligencia artificial y describir su construcción, su funcionamiento, su utilidad, etc., se analiza el papel que pueden tener en el futuro (y presente, ya) de la Informática.

NOTA:
Ediciones Siglo Cultural, S. A., se reserva el derecho de modificar, sin previo aviso, el orden, título o contenido de cualquier volumen de esta colección.

La aparición y posterior difusión del sistema operativo UNIX supuso una revolución en el mercado, de tal modo que se ha convertido en el estándar de los sistemas multiusuario. Su aparente complejidad podría provocar, en principio, un primer rechazo, pero debido a su potencia se convierte rápidamente en una extraordinaria herramienta de trabajo apta para cualquier tipo de aplicaciones.

En este libro se abordan los conceptos fundamentales del UNIX, así como las órdenes más frecuentemente utilizadas. De esta forma el lector se introducirá de manera progresiva en el conocimiento de este potente sistema operativo.